2004

Algorithms for autonomous tandem operation of a dual M113 system

Jared L. Dinkel
Michigan Technological University

ALGORITHMS FOR AUTONOMOUS TANDEM OPERATION OF A DUAL M113 SYSTEM

By

JARED L. DINKEL

A THESIS

submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

MICHIGAN TECHNOLOGICAL UNIVERSITY

2004

Abstract:

In mid-July 2003, the U.S. Army Tank-Automotive & Armaments Command (TACOM) performed a series of experiments at Keweenaw Research Center (KRC), with a remote operated mine roller system. This system, named *Panther Lite*, consists of two M113 Armored Personnel Carriers (APC's) connected by a Tandem Vehicle Linkage Assembly (TVLA). The system has three sets of mine rollers, two of which are connected to the front of the lead vehicle with one set trailing from the trail vehicle. Currently, the system requires two joystick controllers. One regulates the braking of the tracks, throttle, and transmission of the lead vehicle and the other controls the braking and throttle of the rear vehicle. One operator controls both joysticks, attempting to maneuver the lead vehicle along a desired path. At the same time, this operator makes compensation maneuvers to reduce lateral loads in the TVLA and to guide the rear mine rollers along the desired path.

The purpose of this project is to create algorithms that would allow the slave (trail) vehicle to operate using inputs that maneuver the control (lead) vehicle. The project will be completed by first reconstructing the experimental data. Kinematic models will be generated and simulations created. The models will then be correlated with the reconstructions of the experimental data. The successful completion of this project will be a first step to eliminating the need for the second joystick.

Acknowledgements

This project represent one of the biggest ventures I have ever endured. There have been many people who have supported me along the path to completion. First, I would like to thank my research advisor, Jason Blough, and my project advisor, Scott Bradley. Their guidance, knowledge, and understanding have inspired me to make this the best I can make it.

I would like to thank my remaining committee members, Gordon Parker, Thomas Grimm, and Gilbert Lewis for their efforts in this adventure. I would like to give a special thanks to Russ Alger and Jay Meldrum for helping me choose graduate school.

I would like to express gratitude to my friends for supporting me and stimulating my creativity.

Finally, I would like to thank my family. Joe and Amanda, your support and inspiration have helped me strengthen my resolve. My parents have been the best teachers I have ever had and their continued support has made all this possible. Mom and Dad, this is for you!

Table of Contents

Table of Contents (cont…)

Figures

Figures (cont…)

Tables

Chapter 1. Introduction

1.1 Introduction

Military technologies are developed out of a need to increase the safety and effectiveness, both tactically and financially, of the United States Armed Forces. It was through this necessity that TACOM developed the *Sabre* system as a low-cost lightweight alternative to larger single-vehicle applications. This system opened several opportunities that allowed readily available vehicles to perform functions that were normally reserved for a select few vehicles specific to those unique tasks. These few vehicles are expensive and restrictive in terms of mobility and transportation. The *Sabre* is a system created by attaching two M113 tracked vehicles together by a special linkage. This allows the rear vehicle to add its power to the lead vehicle to perform functions that cannot usually be done by a single M113. The *Sabre* is a platform for special application systems.

One of these applications is mine clearing. Anti-personnel and anti-tank mines are potential threats to human life and military property. The *Panther Lite* system was developed to combat these threats. The system is a remote-operated version of the *Sabre* platform and is mounted with battalion countermine roller sets. The *Panther Lite* system requires two human controller inputs. These inputs can be controlled with two people, one controlling each input, or one person controlling both inputs. Both setups have drawbacks. Two individuals require complicated communication as well as the expense of needing the secondary worker. A single individual requires extra concentration to control both vehicles simultaneously.

1.2 Project Outline

The final result of the *Panther Lite* automation project will be a series of functions that lead to the autonomous control of the rear vehicle. A block diagram of this series of functions is shown in **Figure 1.1**. The dashed-line box contains the parts of the process that I developed, of which are described in this report.

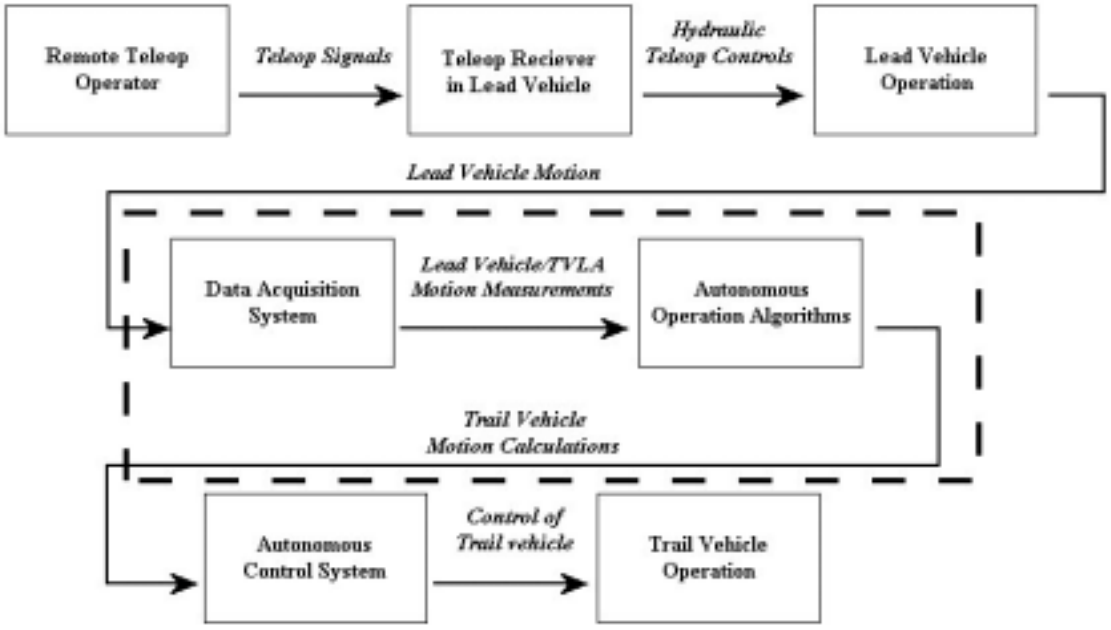

Figure 1. 1: Block diagram of *Panther Lite* automation process. Functions within the dashed-line box are developed in this report.

This report is the first step towards creating a control system that maneuvers the trail vehicle in an intelligent manner, thus removing the need for the second controller. The goal is the development of the algorithms that will drive that control system. Based on kinematic models, these algorithms are developed with the continued purpose of reducing stresses in the Tandem Vehicle Linkage Assembly (TVLA). The algorithms will have modifiable variables that will allow flexibility for future path-coverage development as well as potential component alterations.

My part in the generation of these algorithms was extensive. The next two paragraphs detail my part of the project. For the testing process, the available data acquisition resources were analyzed. From these the best data acquisition system (DAQ) was chosen based on what was decided to be the most useful information. The system was then installed. Special mounting brackets were designed to mount the brake lever string potentiometers within the engine compartment (described in **Section 3.1.2)**. During testing, my role was to control the DAQ and recorded all of the data sets.

All of the data analysis was conducted using Matlab R6.5. Each custom Matlab program discussed in this report was created by myself. My part also included the derivation and development of the all the algorithms and correlation procedures with the assistance of the references listed.

1.3 Assumptions

In order to make this project feasible, several critical assumptions had to be formed. Correction for each assumption would require an entirely new set of tests, calculations, and conclusions. The following is a list of assumptions and a brief discussion of each:

- *Vehicle paths are two-dimensional*. All of the experimental tests were conducted on relatively flat ground. Because of this, there was little variation in the vehicles' vertical bearings. Therefore, the system will be assumed to move only along the horizontal frame and all rotations are about their vertical axes. The consequences of "non-flat ground" would be inaccuracies in the

speed and yaw rate measurements. The measurements would be have an error equal to the sine of the vertical angle of the ground.

- *Individual M113 mass properties are consistent with M113A3 documentation*. The M113 is one of the most used and most widely modified military vehicles in the world. The mass properties, therefore, vary from vehicle to vehicle. The mass properties listed in the M113 transport guidance technical manual, (see **Reference [1]**) will be used in all calculations. Potential variations and the validity of this assumption will be covered in **Section 2.1.2** and **Section 5.2**.
- *The lateral center of mass lies on the vehicle centerline*. The mass properties listed in the M113 transport guidance technical manual did not have a lateral center of mass location. The powerplant on the M113 is located off-center to the right side of the vehicle, as will be explained in more detail in **Section 2.1.2**. The vehicle's centerline will be used in all calculations and this assumption's validity will be discussed in the same section.
- *Uniform ground characteristics for each test run*. The sprocket on a tracked vehicle moves the track and, therefore, the vehicle, such that the top part of the track is moving in the direction of the vehicle's motion at twice the vehicle's speed. The bottom part of the track is in contact with the ground and thus, has little to no velocity. The little velocity it may have is called trackslip. Trackslip is a function of many variables including vehicle mass, velocity, acceleration, turn rate, and ground consistency. When the algorithms are put into a control system, a variable relating specific ground types must be

also be implemented. This will be covered more in **Section 3.3.5** and **Section 5.2**. For the purposes of this project, it will be assumed that ground characteristics remain uniform within individual test runs. It is, however, understood that different test runs are in different locations and thus, have different ground properties. The consequence of variation in ground characteristics within the a test run would create inconsistency in trackslip characterization.

Chapter 2. Background

2.1 M113 Family of Vehicles

2.1.1 Vehicle History

The M113A3 is the main vehicle used in the *Panther Lite* project. The following section is a summary of information provided by **References [1]** and **[2]**. Introduced in 1960, the M113 was a breakthrough Armored Personnel Carrier (APC) created by Food Machinery Corporation (FMC). Air transportable, low-velocity air-droppable, and able to move over water (at a speed of 4 knots), the M113 could carry up to eleven armed soldiers as well as the driver and track commander. This was a platform that could carry troops in an armored shell over a battlefield with difficult terrain. The air transportable and air droppable qualities of the M113 increased the range of combat and supported "rapid deployment" scenarios. So successful was this vehicle that it began its own family of vehicles. Over 40 variants have been identified including the M577 Command Post Carrier and the M1064 Self-propelled 120mm Mortar.

In 1964, the M113A1 upgrade package was developed. This package included a 212 horsepower diesel engine that replaced the old gasoline engine. The package could be installed into an old M113 and was automatically installed with the new vehicles being built, now under the M113A1 name. This upgrade package also spawned a line of unarmored variants such as the M548 Cargo Carrier and the M667 "Lance" Missile carrier. These unarmored versions were successful for non-direct-combat.

During the mid-70's, government officials concluded that the current vehicle was in need of more improvements. In 1979, the M113A2 package was

developed. This package featured enhancements in both cooling and suspension. Torsion bars were stiffer, allowing for greater ground clearance and an extra shock absorber was added to the second road wheel on each side. The fuel tanks on the M113A2 were moved externally and armored. This freed up 16 cubic feet of cargo space on the interior. This did, however, increase the overall weight of the vehicle. Because this extra weight reduced the vehicles' freeboard when afloat, the swimming requirement of the vehicle was lifted. Swimming with the M113A2 during training was then banned. Once again, the package increased the M113 family of vehicles (FOV).

The most recent upgrade was introduced in 1989. It included vast improvements over the previous model. A newer turbocharged diesel engine increased the vehicles power to 275 horsepower. A Kevlar lining was installed in the crew compartment as extra protection in case of armor penetration. As with the M113A2, the fuel was stored in the external fuel tanks on the M113A3.

As of September of 1998, the M113 FOV made up 46% of the vehicles in the U.S. Army armored division. Nearly half of those are M113's. The next upgrade is scheduled for 2006 and would include mostly computer-based enhancements. At this time, there are no foreseeable replacements being produced or developed, so the M113 program will continue far into the future.

2.1.2 Vehicle Description

The M113A3 is a front sprocket drive vehicle and has two track systems. Each track is made up of either the T130E1 or the T150 track shoe. Both shoe types are center guide, single pin metal shoes with detachable rubber pads. Due to the forward offset of the right sprocket, the right track has sixty-four shoes, while the left track has

sixty-three shoes. Each track system has five equally spaced roadwheels covering 105 inches by 15 inches of ground area. The roadwheels are individually supported by torsion bars for suspension. Shock absorbers are installed on the first, second, and fifth roadwheels. An individually adjustable idler tensions each track. The M113 track systems do not have support rollers. After the suspension upgrade included with the M113A2 package, the vehicle had 15 inches of clearance. **Figure 2.1** shows profile drawings of the M113A3. The external fuel tanks have been omitted, but the overall

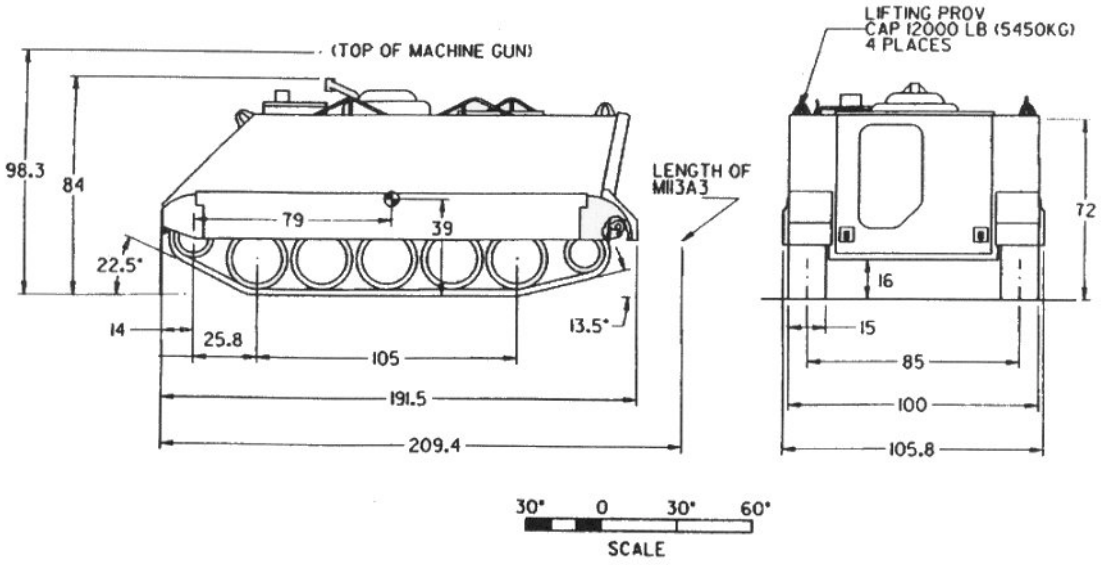

Figure 2. 1: M113A3 and M113A2. All dimensions are in inches.

length measurement shows the length with the tanks.

Figure 2.1 shows that the center of mass of an M113 is located 93 inches behind the front edge of the hull. The vehicles used in the *Panther Lite* system will be operated via remote control units that utilize hydraulic pumps to control the vehicles. They will also have partially supported mine rollers attached, as well as the Tandem Vehicle Linkage Assembly (TVLA). Therefore, depending on where everything is installed internally and how much the external attachments are supported by the M113,

the center of mass may shift several inches. Recommendations for this are located in **Section 5.2**.

A test measuring weight per track was used to determine the vehicles lateral center of mass and determine the legitimacy of declaring the centerline an acceptable approximation. Keweenaw Research Center engineer, Chris Green, conducted this test in May 2003. The results of this test can be found in **Table 2.1**. An initial

Lead Vehicle				
	Left Track	15,070 lbs.	Right Track	15,550 lbs.
Trail Vehicle				
	Left Track	10,490 lbs.	Right Track	11,380 lbs.

Table 2. 1: Vehicle weight distribution for each track system.

discrepancy was that the total weight of the trail vehicle was less than 22,000 pounds. This was due to the trail vehicle being completely empty when tested. During actual operation, the trail vehicle will have ballast weight. Nonetheless, with a distance of 85 inches between the centerlines of the two tracks, the center of mass of the lead vehicle is only 0.6 inches, or 0.05 feet to the right of centerline. The center of mass of the trail vehicle is only 1.7 inches or 0.14 feet to the right of centerline. These values are very small and the error resulting from the variation from centerline would be a small fraction of the values themselves and would fall within a resonable error margin. Therefore, the centerline of the vehicles can be assumed to be a viable location for the center of mass.

The M113 is steered using a system of brake clutches. The vehicle is propelled through a transmission that includes three forward gears. The system was in first gear for the entire test period described in **Section 3.2**. When the vehicle is in gear, both tracks are powered using the throttle pedal. Steering is controlled by slowing down the tracks independently using brake sticks. This technique is called “skid steer.” The

brake sticks are connected by mechanical linkages to a series of clutches on the drive axel that adds braking force to either track. These linkages will be used to measure system input. (See **Section 3.1.7**.)

2.2 M113 Saber System

2.2.1 Description

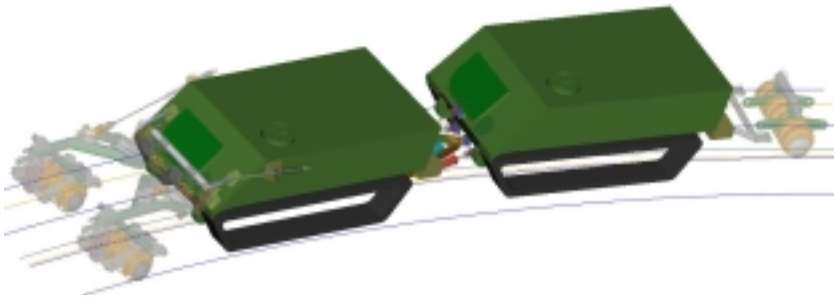

Figure 2. 2: ProEngineer *Saber* model. (Translucent parts are part of the *Panther Lite* system and were not attached during Tacom testing, 07-2003.)

The *Saber* system was developed to allow two M113A3s to work together to perform a function reserved for a larger vehicle. The term *Saber* was adopted to describe any system that is composed such as this. The testing described in **Section 3.2** was done with a generic remote *Saber* system. This means that with the exception of the remote control components, the system consisted of only two M113A3s and a TVLA. There were no other components, such as rollers, added to this vehicle.

2.2.2 Panther Lite

The *Panther* system is an existing method of mine clearing. The *Panther* consisted of little more than an M60 chassis, without a turret, and a mine-clearing device. These devices came in many forms, from mine plows to chains that slam into the ground to heavy metal rollers. One such roller set was called the battalion countermine roller set

and will play a crucial part in the *Panther Lite* system. The *Panther* was big and heavy. It could only be transported by air via the C5 *Galaxy* and it was not air droppable.

When it was determined that a replacement had to be developed, the *Saber* system was considered. Not only is the *Saber* system smaller, lighter, and air droppable, but it is also transportable via smaller aircraft such as the C130 *Hercules*. In addition, since the *Saber* system uses M113's, standard versions may be modified in the field in an emergency. When a battalion countermine roller set is attached in front of each track of the lead vehicle and a modified set of rollers cover the gap between the two, the *Saber* system becomes the specialized *Panther Lite* system. Named for being a lighter version of its predecessor, the *Panther Lite* is prepared to replace the *Panther* as the mine clearance method of choice.

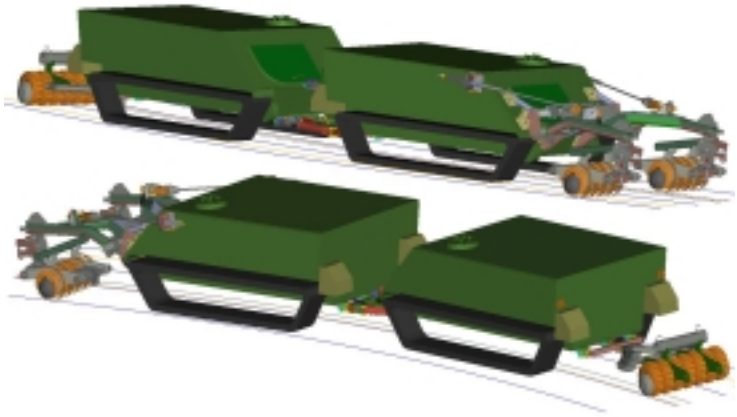

Figure 2. 3: ProEngineer *Panther Lite* model.

2.3 TVLA

When TACOM began to develop the *Sabre* system, Keweenaw Research Center was contracted to develop the tandem vehicle linkage assembly to join the two M113A3s. The development of the TVLA was U.S. Army contract number DAAE07-00-L05

(WD011). The TVLA had to be strong enough to handle the stresses imbued by the M113A3s and any potential components that may be installed in the system. The result was an A-frame design that connected the rear of the lead vehicle via three uni-axial horizontal hinges.

Figure 2. 4: ProEngineer model of the TVLA. The hydraulic manifold is the light blue object.

The TVLA is connected to the trail vehicle at the narrow end of the A-frame by means of a large universal joint. This setup created a vertical pivot point between the lead vehicle and the TVLA as well as between the TVLA and the trail vehicle. It also created only one horizontal pivot between the TVLA and the trail vehicle. Therefore, given the first assumption of two-dimensional motion, the lead vehicle and TVLA can be considered a rigid unit.

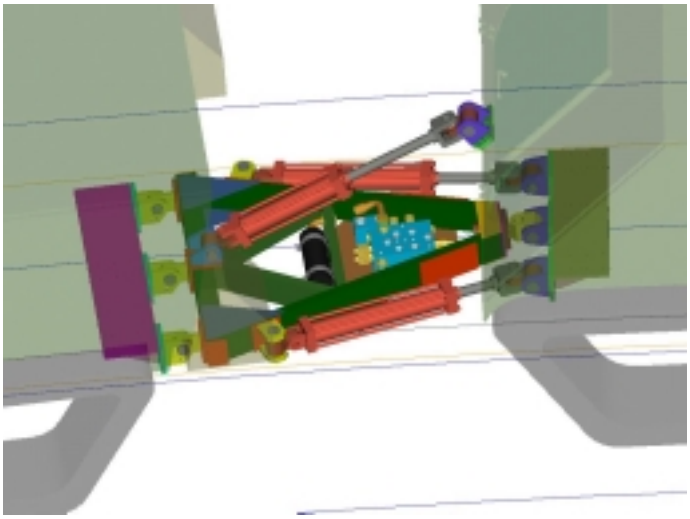

Figure 2. 5: ProEngineer model of the installed TVLA. The lead vehicle is on the left side and the trail vehicle is on the right.

To add extra stability, two heavy-duty, horizontal, hydraulic cylinders were added connecting the TVLA to the trail vehicle. Another cylinder was angled vertically from the top centerline of the TVLA to the trail vehicle. These cylinders were connected to an adjustable hydraulic control manifold. By adjusting the relief valves, the horizontal pivot could be made more or less stiff. The cylinders provide a potential measuring point for vehicle orientation, as will be discussed in **Section 3.1.2**.

Testing was conducted at Keweenaw Research Center between July 7 and July 16, 2003. The purpose of the tests was to determine the effectiveness of a remote operation system. Though the remote operation system was designed for the *Panther Lite* project, the vehicle test setup only contained the two M113s and the TVLA. There were no rollers, carriages, or support arms. The testing will be explained in detail in **Section 3.2**.

Chapter 3. Experimental Testing

3.1 Data Acquisition

This algorithm generation project was pitched two weeks before the two-week July 2003 testing period. This left very little time to investigate the scope of the project and determine what measurements were needed on the system. Therefore, measurements were made on as many characteristics as possible, even if they seemed irrelevant.

3.1.1 Datron

One of the limiting factors was the data acquisition system. The best available system was the Datron AEP-2 (Acquisition Evaluation for PC). A compact modular designed DAQ system, the AEP-2 is a dedicated platform for on-board testing and the system power is compatible with a vehicle's 12-volt battery. One convenient aspect of the AEP-2 system is the DAVIT-bus, which allows for a data link between the Datron and the serial port on a computer. This allows the system to be controlled via a specialized program on a PC.

The system had thirty-two differential analog channels each independently adjustable from ±1 volt to ±15 volts in 1-volt increments. Six digital counters allowed pulse quantity measurements. Three inputs were reserved for strain gage inputs. The system had a sample rate of 10 milliseconds or 100 Hz. Due to the low speeds the vehicles are expected to maintain, analysis of vibration characteristics is not expected. Therefore, the sample rate should be more than adequate for the requirements of the project.

The Datron program is highly customizable. Carl Keranen, a research associate at KRC, developed a unique program for this test setting. This program recorded all channels as will be described in the following sections as well as displayed each channel real-time during each test run. Each set of data was given a unique name and then after the tests, all of the data could be retrieved in the form of comma separated variable (CSV) files. These files were loaded into Microsoft Excel or Matlab for processing. Also within this program, calibration values could be entered, such as volt/degree for the roll, pitch, or yaw measurements from motion pack. This allowed the CSV file values to be in calibrated units.

3.1.2 String Potentiometers

A string potentiometer, also known as a string pot, is a displacement transducer. Based around a linear variable resistor, a string pot makes use of Ohm's law:

$$V = I \times R \tag{3.1}$$

The basic schematic of a string pot is shown in **Figure 3.1**.

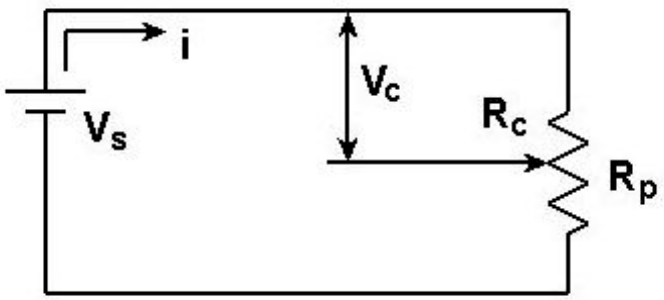

Figure 3. 1: Basic schematic of a string potentiometer.

Given a specific voltage source, V_s, of 5 volts, the voltage loss, V_c, between the variable resistor common and contact is determined by the resistance, R_c, and the current flow, i. The current remains the same no matter where the contact is. This current is, therefore a known quantity that is calculated using Ohm's law (Eq. 3.1) from V_s and the total resistance of the potentiometer, R_p. The contact resistance, R_c, can then be calculated using the following equations:

$$R_c = \frac{V_c R_p}{V_s} \tag{3.2}$$

Notice that the contact resistance is linearly dependent to the contact voltage, V_c.

A string is connected to the contact. This gives the string pot its name. The string is spring-loaded to retract into the transducer. As the string is extended, the contact increases the contact resistance. This increase is linear and the electrical technicians at KRC measured the coefficient during a caliper calibration. These coefficients were entered into the Datron test program as necessary. String potentiometers produced analog signals.

3.1.3 Optical Encoders

It was necessary to measure the speeds of all four tracks. At several points in the system, speed needed to be measured. Made by Correvit, optical encoders are used to accomplish this by measuring rotation of the drive sprocket, which is directly related to the track speed. Incorporating a flexible driveshaft that is connected to the face of the sprocket, the encoders create a pulse signal of 1000 pulses per revolution. The calibration factor, *cf*, can be calculated using this value and the effective radius, r_e, of the drive sprocket according to the following equation:

$$cf = \frac{2\pi r_e}{1000\ pulses/rev} \qquad (3.3)$$

The calibration factor can then be entered in to the Datron program. The Datron system uses the counters to input the pulse signals.

3.1.4 5th Wheel

The 5th wheel is another way to calculate vehicle speed. Consisting of a 600 pulse per revolution (ppr) optical encoder, made by Labeco, connected to a bicycle tire, the 5th wheel is designed to measure the speed of the ground relative to a vehicle. The tire and encoder are connected to a support frame via a simple spring/shock absorber suspension. This frame can then be attached to the surface of a vehicle. Like the encoders described in **Section 3.1.3**, the 5th wheel produces a pulse signal and is read using the Datron signal. The calibration factor for the 5th wheel is 86.9565 pulses per foot, calculated by dividing the 600 ppr by the tires circumference.

3.1.5 Motion Pack

The Systron Donner motion pack is actually six transducers in one. Six internal accelerometers measure three orthogonal accelerations as well as roll, pitch, and yaw. The system was meant to give precise vehicle motion. However, there was an error in the program that added an unknown offset to the linear accelerations. Fortunately, there was no such error in the rotational velocities. A third party company that specializes in calibration of such devices calculated the correction factors for converting from voltage to g's of accelerometer. Motion pack signals were measured as analog signals.

3.1.6 Pressure Sensors

It was determined that pressures within the hydraulic system of the TVLA may give a qualitative measurement of how well the entire system was performing. For this purpose, two Sensotec pressure sensors were selected to make such measurements. The sensors are appropriately named strain-type sensors due to the fact that the signal created behaved like a strain gage. Therefore, the strain gage inputs in the Datron system can be used to record these signals.

3.1.7 Transducer Location

The lead vehicle of this system contained the remote control systems for the vehicles, including high-pressure hydraulic pumps for its linear actuators. The lead vehicle also was used to power this control system as well as the camera setup mounted on the vehicles. Because of this, it was decided that the Datron system be located in the rear vehicle. This not only allowed plenty of room and power for the system, but also allowed extra room for myself, the operator of the DAQ system.

Figure 3. 2: Mounting bracket for 2” string pots on the lead vehicle.

There were a total of six potentiometers used in the experiment. Two 2-inch string pots were mounted in the engine compartment of the lead vehicle. These string pots were used to measure the lever arms connected to the brake stick levers and going into the clutch-pack-braking differential. The purpose of these string potentiometers was to measure input into the system. A special bracket built from an 8-inch C-channel was used to place the string pots in position to most effectively measure the lever motion. The bracket was mounted directly to the top of the differential casing.

Figure 3. 3: Mounting bracket for 10” string pots on the trail vehicle.

Likewise, two similar 10-inch string pots were mounted on a matching bracket on the trail vehicle. There were only two 2-inch string pots and two 10-inch string pots

available for use. Therefore, it was decided that both of the 2-inch string pots would go into the lead vehicle and both of the 10-inch string pots would go into the trail vehicle. The 10-inch string pots are externally identical to the 2-inch string pots. They perform the same function, to measure the braking inputs to the trail vehicle. The wiring for the two string pots in the lead vehicle was run through the back of the engine compartment and out to the Datron system in a manner so as to not cut or bind the cables..

The other two potentiometers were mounted on the TVLA. A 24-inch string pot was mounted along the vertical hydraulic cylinder. The purpose of this was to measure the vertical angle between the two vehicles. The last string pot was a 5-foot model mounted to the left horizontal cylinder. This allowed the measurement of the horizontal angle between the two. The cabling for the two string pots were attached up to the front of the trailing M113 as were the other cables.

Figure 3. 4: Wheel speed monitor attached to the right sprocket of the trail vehicle.

Four optical encoders measured track speed. One encoder was attached to the sprocket of each track system. Specially built brackets mounted to pre-existing holes in the hulls of the M113s supported the encoders. The flexible driveshaft bolted directly to the outside face of the sprocket. Wiring for the lead vehicle encoders ran down the sides

of the vehicle, and combined with the string pot cables. The rear encoders' wiring ran up the side of the vehicle to the Datron system.

Figure 3. 5: 5th wheel mounting position. Rear end of the trail vehicle.

The 5th Wheel attached off the left side of the trail vehicle to pre-existing holes drilled in its hull. See **Figure 3.5**. Wiring ran onto the top of the rear vehicle and down through the port to the DAQ system.

Figure 3. 6: Installed pressure sensor with electrical tape protecting data connections.

As stated in **Section 3.1.6**, pressure sensors were selected to monitor the hydraulic pressures in the TVLA. The hydraulic manifold was designed with measuring ports. These allowed the hydraulic pressure to be monitored efficiently with no alterations to

the system itself. The pressure sensors' wiring followed the same path of the 10-inch string pots to the Datron system.

3.2 Testing

TARDEC Vehicle Sensor Integration Group requested assistance from the Joint Programs Office (JPO) Unmanned Ground Vehicles for integrating a Legacy Omnitech Teleop Kit onto the Panther Lite. JPO, with the help of Titan System Corporation, designed the teleop system. The purpose of the tests was to integrate the system into the *Panther Lite* system.

3.2.1 Involved Personnel

The test was a collaboration between a group from TACOM and several KRC personnel. The test coordinator was Andy Vogeli of TARDEC Vehicle Sensor Integration Group, VSIG, from TACOM. Also from TARDEC VSIG was Keith Kamysiak, who was the computer specialist. He was in charge of setting up the camera network on the vehicles and control program display. During testing, Kamysiak documented the experiments via video recording with audio narration. Andrew Smith was the remote control operator from Titan Systems Corporation through Robotics Systems/Joint Programs Office, RS/JPO. Also from RS/JPO, was Mike Moeller. Moeller was the driver of the lead vehicle. When the remote control was not in operation, the drivers manually controlled the vehicle. This was done mainly to move the system from the main facility to the test areas. While the vehicles were in remote control operation, the drivers' functions were to communicate information and control the emergency stop

mechanisms. It was permissible for the vehicles to be manned, as there was no blast testing for these experiments. Vehicle occupants are not permitted for blast tests because of the danger. Chris Green, a KRC senior research engineer, initially drove the trail vehicle. After the first two days of testing, the aforementioned Carl Keranen took over as driver. Dave Fredrick was the TARDEC VSIG supervisor. Supervising for the JPO Unmanned Ground Vehicles was Danny Price. Scott Bradley was the KRC supervisor. As mentioned in **Section 1.2**, my part in the testing included controlling the data acquisition software, recording the data sets, and communicating related information to the aforementioned team members.

3.2.2 Vehicle Setup

The system tested was a *Saber* system, the *Panther Lite* without the mine rollers or mine roller attachments with some minor differences. Instead of an M113A3, the lead vehicle was an M113A2 supplied by JPO. The rear vehicle was an M113A1 supplied by KRC. Though the vehicles were not M113A3s, the differences in operation would be minimal. For easy designation during testing, a large "I" was marked in duct tape on both sides of the lead vehicle and a large "II" was marked on both sides of the trail vehicle.

Both the M113A1 and the M113A2 are controlled via brake sticks, whereas the M113A3 has a yoke. This has no effect on the remote control systems. The hydraulic actuators that control vehicle braking and turning are connected to linkages in the engine compartment. These linkages are the same for all models of M113.

Three cameras were installed on the system. These allowed the controller to navigate the system as well as observe the condition of the TVLA, and all three video feeds were visible on the control program at all time. One camera was mounted on top of the lead vehicle on the centerline. The cameral pointed forward and was the main navigation camera. In order to assist with orientation, a small pole was installed several inches directly in front of the camera, with the top portion painted. The purpose of the pole was similar to that of the sights on a gun. The distance was determined from the front of the vehicle to the spot on the ground that the top of the pole "pointed at". This gave the controller a defined point of reference.

Figure 3. 7: JPO M113A2 with "I" designation on side of hull.

Figure 3. 8: Forward facing camera of lead M113 and centering pole.

Another camera was mounted on top of the lead vehicle on the right-rear quarter. This camera was positioned to observe the entire TVLA. The third camera was mounted on top of the trail vehicle on the left-rear quarter facing rearward. The purpose of this camera was to observe everything behind the system. It also allowed the controller to determine if cones were knocked over during the tests that required path motion. Lastly, the remote transceiver was installed on the top right-rear quarter of the lead vehicle. This antenna sent system information to the controller and received command functions for the control system.

Figure 3. 9: Complete vehicle setup before application of numerical designators. Note: The transceiver is secured for travel.

3.2.3 Control Setup

The original test plan included multiple control scenarios relating to the vehicle system and the command vehicle. The command vehicle (CV) was an M998 HMMWV. The vehicle was outfitted with a transceiver matching the one on the *Sabre* system.

The first scenario involved a static CV while the *Sabre* was maneuvered through an open-field refered to as the "spin-up, packed snow area", SUPSA. Though not actually covered with snow, this area was an open flat field, half of which was composed of crushed asphalt. The other half of the field was a softer grass covered soil. This was a breaking in period that allowed the controller to get used to the system. For the purpose of this paper, the term "controller" will refer to the person operating the remote system. This period presented the group with an opportunity to check the system for problems before the critical tests were conducted. Later in testing, the scenario was repeated with a mobile CV. However, since the goal of this project is to correlate algorithms with viable

experimental data, it was decided not to acquire data from the mobile CV test due to the greater chance for extraneous commands to be input into the system.

The remote communication was tested in three more scenarios. All three were conducted with the CV stationary. The first two involved driving the *Sabre* behind trees and then hills in order to determine what would disrupt the communication. The system would shut down if communications were interrupted. The third scenario in this series was a maximum communication distance test. This involved driving the *Sabre* in a line-of-sight path away from the CV until communication was disrupted. Due to the fact that the successful completion of these scenarios involved the disruption of the communication of the controller and the shutdown of the vehicle, data were not acquired.

When the controller had become more comfortable with the system, handling loops were constructed. These were first run with a static CV and then repeated with the mobile CV. More in-depth descriptions of these handling loops are described in **Section 3.2.5**. As with the open-field maneuvers, data from the mobile CV tests were not acquired.

3.2.4 Test Schedule

The test was scheduled to take two weeks to complete, from Monday, 07/07/03 to Friday, 07/18/03. The first week's agenda was to prepare the vehicle setup. This included drilling all necessary holes for the TVLA, attaching the TVLA, and installing the teleop system. Also included during this time was the installation of the instrumentation and DAQ system. The second week was scheduled to contain all of the tests. Monday and Tuesday were intended to be the static CV tests and the Wednesday

and Thursday, the mobile CV test. This would allow the controller the necessary time to become used to the system before attempting control in the unstable mobile environment.

The vehicle setup and installation began on schedule. The groups involved proved efficient enough to finish this stage by the end of Thursday, 07/10/03. The morning of the following Friday was spent fine-tuning the system. By afternoon, the system was moved outside. A comprehensive check of the safety systems followed. It was critical to assure guaranteed safety. There were no problems found on the safety inspection, so the system was moved manually to the SUPSA for testing.

3.2.4.1 Friday, July 11, 2003

The testing began with a safety check. This safety check would be repeated anytime that the vehicles were turned off for any length of time. The controller would begin the check by testing the emergency stop that was part of the control program. After the emergency stops were reset, the observer would test his emergency stop. The observer gave a third person aspect to safety, as he was not involved in the operation of the vehicle. Andy Vogeli functioned as the observer for the tests. Again, after all emergency stops were reset, the driver of the lead vehicle would trip his emergency stop. Lastly, the driver of the trail vehicle would test his emergency stop. Each safety check successfully passed without incident.

Tests conducted on this day were considered to be a “warm-up” period for both the system and the controller. The CV was parked on the east border of the SUPSA in order to stay clear of the *Sabre* maneuvers. The maneuvers consisted of movements along non-predetermined paths. These paths included combinations of turns. A total of nine measurements were made during this time. Each measurement was of the form,

Sabre#, where the # symbol designated the test number. This numbering convention was repeated for each separate day of testing where the number reset each day.

During these tests, the most prominent problem the controller had was the inability to maneuver the vehicles out of a turn. The system had a tendency to keep the system locked at the angle needed for the turn and the vehicles would have to give individual effort to straighten themselves out. Otherwise, the lead vehicle would use the TVLA to jerk the trail vehicle sideways off of a turn. It was hypothesized that this may be due to too much restriction of the fluid flow between the hydraulic cylinders, caused by having the relief valves set too high. Part of the way through the test designated *Sabre4*, the string end of the string pot attached to the left hydraulic cylinder on the TVLA came loose. It was secured before the *Sabre6* test was conducted. Therefore the *Sabre4* and *Sabre5* tests were immediately labeled as unusable data for any calculations that include cylinder displacement.

At the end of the day, it was determined that none of the data were acceptable for use in calculations. This was due to the fact that the test added high stresses to the TVLA, the hydraulic cylinders, and all attachments. This was not the quality of data that was desired for correlation. It was also determined that the solution to the TVLA stiffness problem must be resolved before the more intensive tests began the next week. Therefore more testing was conducted on Saturday, 07/12/03.

Test Data – Friday, 07/11/03	
Sabre1	Driving out to the SUSPSA.
Sabre2	Random driving on the SUPSA.
Sabre3	Random driving on the SUPSA.
Sabre4	Random driving on the SUPSA.
Sabre5	Random driving on the SUPSA.
Sabre6	Random driving on the SUPSA.
Sabre7	Random driving on the SUPSA.
Sabre8	Random driving on the SUPSA.
Sabre9	Random driving on the SUPSA.

Table 3. 1: Summary of test data from 07/11/03.

3.2.4.2 Saturday, July 12, 2003

The sole purpose of this day of testing was to resolve the TVLA problem. As stated earlier, it was hypothesized that the high pressure setting on the relief valve caused the stiffness problem. To correct this problem, the relief valve was slowly opened, incrementally. After each increment, the system would be driven via remote control to assess the effect of the adjustment. It was only when the relief valve was completely open that the controller felt satisfied with the response of the system. For the rest of testing the relief valve was left in the open position. Due to the inefficient nature of the system behavior in the tests on this day, no data were recorded.

3.2.4.3 Monday, July 14, 2003

This day marked the beginning of the testing that was intensive on validation of the teleop system. Testing began with free runs of the system with a static CV to reacquaint the controller with the system. Then the interference and maximum distance communication specific testing was conducted. There was no data acquired on this date.

3.2.4.4 Tuesday, July 15, 2003

The first of the beneficial data were recorded on this date. However, before the testing, a newly positioned camera setup had to be measured and tested. Following this, the first set of runs consisted of "warm-up" free runs, the act of which at this time became known as "meandering". Then, several handling loops were set up by the TACOM group for static CV testing. The first series was of a narrowing cone test. This test was repeated five times. The goal for this test was for each consecutive attempt, to complete it faster and more accurately. More detailed information on the special maneuvering loops can be found in **Section 3.2.5**.

The second series was a slalom run. This test was repeated three times. Like the narrowing cone test, the goal of the slalom tests were to consecutively improve on speed and the quality of the turning. On the first run of this test, extra cones were skipped at times. The second run did not skip any cones, but turns were very sharp, such that when the system reached the coneline, its path was nearly perpendicular to that line. The third run was much faster with better, less angled turns.

Finally, a simple high-speed test was conducted. This was over rough terrain, so the 5th wheel speed accuracy was questionable. It was also in the SUPSA, which is too small of an area for the system to reach its full speed. A total of eleven sets of data were acquired, however, the first set, *Sabre1*, had an error while acquiring, so the data were corrupted and could not be uploaded to the PC. More testing was conducted on this date, mostly pertaining to mobile CV testing. Therefore, no more data were obtained.

Test Data – Tuesday, 07/15/03	
~~*Sabre1*~~	Data corrupted. Not used.
Sabre2	New camera position test. Meandering
Sabre3	More Meandering.
Sabre4	Narrowing Cone Test. 2x (Circles around first.)
Sabre5	Narrowing Cone Test. 2x
Sabre6	Narrowing Cone Test. (Circles around first.)
Sabre7	Maneuvering to slalom cones.
Sabre8	First slalom.
Sabre9	Second slalom.
Sabre10	Third slalom.
Sabre11	High-speed test.

Table 3. 2: Summary of test data from 07/15/03.

3.2.4.5 Wednesday, July 16, 2003

By this day, testing was well ahead of schedule. The only remaining test that the TACOM group needed to conduct was the high-speed test. Since the high-speed test was a highly stressful test, it was reserved for the end of the testing. Before that, there was time for a test specific to this project. The modified figure eight test was developed and set up. The test layout can be found, described in detail, in **Section 4.5.4**. Initially, when the circuit was first run with the cones lining the inside of the figure-eight (Guide Cone Configuration 1), the controller could not see the cone while turning. As a result, the system path was erratic and many cones were trampled under-track.

To correct for this problem, the cones were all moved to the outside of the figure eight (Guide Cone Configuration 2). This drastically increased visibility and helped with navigation. The circuit was run six times, three times in each direction. With the newer cone configuration, the system's motion through the curves was much faster and smoother. Fewer cones were crushed.

After the figure eight testing, the vehicles were maneuvered manually to what is referred to as the “ice rink”. During winter testing, this area is covered with a large slab of ice for vehicle handling tests. In the summer, however, this area is just a long stretch of flat ground with a slight slope towards one end. This made for a perfect high-speed test location. The CV was parked on the top of a set of slopes next to the ice rink. This location gave the vehicle a good observation point over the system during the test. The test was conducted twice. The first test was moving up the slope and the second test was moving down the slope. After the second run, the system was turned around for a third run. At this time, the TVLA was noticed dragging on the ground. The apex spindle, which allows the trail vehicle to roll separately from the lead vehicle, became clogged with dirt and debris. This caused the spindle to seize and then, with the stresses of the test, break apart. The rut caused by the dragging of the a-frame was used to determine the point of failure. The rut began after the system had slowed down, about halfway through its turnaround for the third run. This means that the *Sabre8* data file is a viable source of information. Also, test data (see **Figure 3.10**) showed that there were no pressure spikes toward the end of the run. During the testing on Friday, July 11 (see **Section 3.2.4.1**), when the TVLA stiffness was too high, there were spikes of between 600 and 800 psi. If the connection had begun to fail during the run, spikes of similar magnitude would have been recorded.

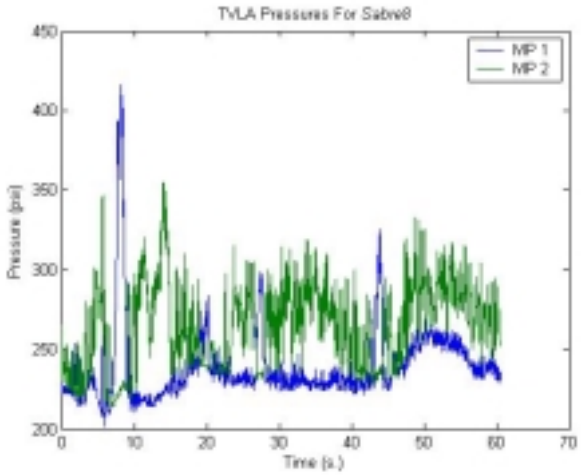

Figure 3. 10: TVLA hydraulic pressures during the *Sabre8* test run.

After the failure, the vehicles were disconnected and manually driven back to the shop for component cleanup and removal. The TVLA was loaded onto a front-end loader and returned as well for analysis. This marked the end of testing for the week. A total of eight data sets were acquired on this date. The most notable are *Sabre3* and *Sabre4*, which were the modified figure eight tests with cone configuration 2. These are expected to be the most beneficial data sets used in model correlation and validation.

Test Data – Wednesday, 07/16/03	
Sabre1	Meandering.
Sabre2	Modified figure eight. Large loop CCW #1
Sabre3	Modified figure eight. Large loop CCW #2,3,4 (#3 @ 200 s.) (#4 @ 365 s.)
Sabre4	Modified figure eight. Large loop CW #1,2,3 (#2 @ 195 s.) (#3 @ 355 s.)
Sabre5	Maneuvering to ice rink.
Sabre6	Maneuvering to ice rink.
Sabre7	High-speed test #1. Up slope.
Sabre8	High-speed test #2. Down slope.

Table 3. 3: Summary of test data from 07/16/03.

3.2.5 Handling Loops

This section covers the handling loops used during data acquisition and their locations. Maneuvers conducted that were not part of the data acquisition process, such as mobile CV and communications tests will not be covered in this section. Where possible, diagrams will be used to assist in understanding the usefulness of these maneuvers. Larger and more in-depth diagrams can be found in **Appendix D**.

3.2.5.1 Free Run

Though not specifically a handling loop, free runs were combinations of basic maneuvers such as straightaways, left and right turns, and s-curves. A nice aspect of a free run is that it allows investigation into the transition between these basic maneuvers. Since the free runs are non-directed maneuvers and vehicle control is under the discretion of the controller, the basic maneuver content of each free run is different.

3.2.5.2 Narrowing Cone Test

The narrowing cone test is a handling loop designed to improve controller familiarity with the system by forcing the controller to navigate a set of narrowing cones. The test was intended to be run multiple times for continuous improvement. The *Sabre* system is parked between two cones separated by 21 feet. This is the "starting line" for the test. Two rows of eleven cones begin 75 feet in front of the system at the start. Consecutive cones are separated by 10 feet for column lengths of 100 feet. The first set of cones is spaced 22 feet apart. Each consecutive set is spaced one foot less apart from the set before. This gives the last set of cones a clearance of 12 feet. Keep in mind that the total width of the vehicles with the 5^{th} wheel attached is slightly less than 10 feet.

The goal for the controller is to continually complete the test run with higher speeds and more control.

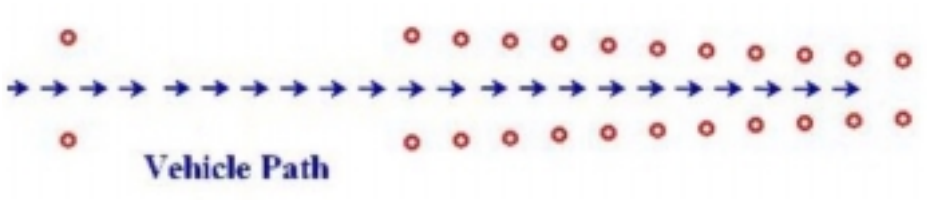

Figure 3. 11: Narrowing cone test setup.

3.2.5.3 Slalom

The slalom is a run designed to test the turning capabilities of the controller as well as the system. The clear definition of each turn makes this test an ideal source of data for this project. Like the narrowing cone test, this handling loop is meant to be performed several times, until the controller becomes comfortable with the behavior of the system. The test was set up by placing eleven cones in a row, each separated by 50 feet. The object of the test was to drive past the first cone on its left side, turn and past the next cone on its right side. The vehicle would continue the sinusoidal motion through the cones. On the first slalom test conducted on July 15 (see **Section 3.2.4.4**), it was necessary for the controller to skip a cone in order to widen the turn. As the controller repeated the test, the turns became smoother, faster, and more efficient.

Figure 3. 12: Slalom cone setup.

3.2.5.4 Modified Figure Eight

The modified figure eight was an attempt to measure the basic maneuvers in one circuit. It was designed to have a relatively small-, medium-, and large-radius turn, as well as a straightaway and an s-curve. The small loop of the figure eight was composed of a circle whose radius was 45 feet. This loop connected to a semicircular arc with a radius of 90 feet. The other end of this arc was connected to another 75-foot arc. This produced a gap between the free end of the 45-foot arc and the free end of the 75-foot arc. This proved to be an ideal location for a straightaway. Therefore, if the circuit was navigated in a CCW route about the large loop, the maneuvers would include the following: Straightaway, medium left turn, large left turn, left-to-right s-curve, and a small right turn. If the circuit was navigated in a CW route about the large loop, the maneuvers would include: Straightaway, small left turn, left-to-right s-curve, large right turn, and a medium right turn.

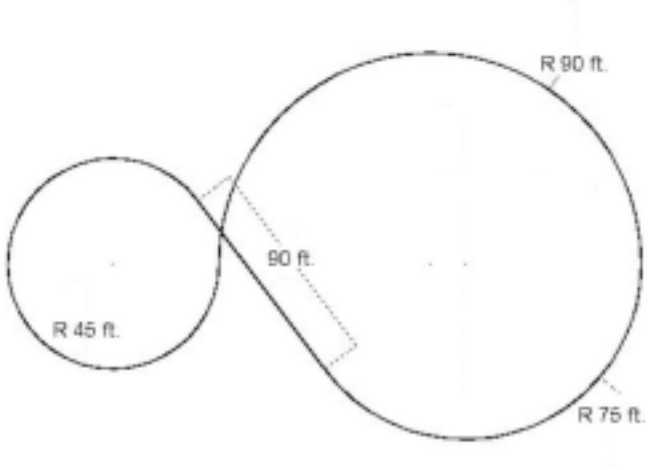

Figure 3. 13: Modified figure eight dimensional layout.

Guide Cone Configuration 1

Initially, during the tests on July 16, (see **Section 3.2.4.5**), it was decided that cones would be placed inside the path. This would make sure that the controller would not turn overly sharp. The cones were placed 7.5 feet from the path. This would result in a lane that was 15 feet wide. Problems arose when the controller was unable to see the cones on the inside of a turn. This was because the navigational camera was pointing straight forward, which is outward when making a turn. Because of this, the first test was slow and lopsided. Many cones were crushed.

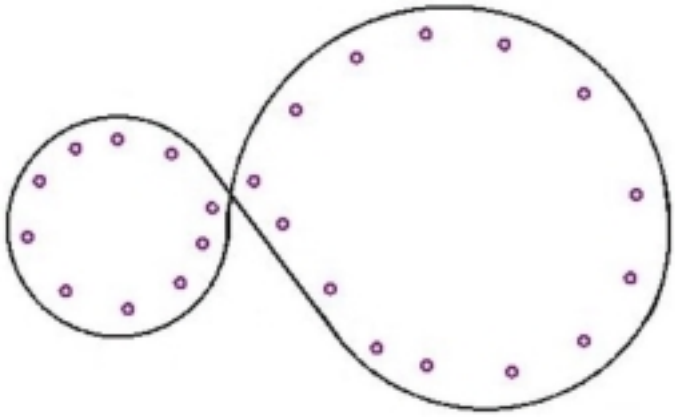

Figure 3. 14: Guide Cone Configuration 1.

Guide Cone Configuration 2

A new configuration placed the cones on the outside of the path. As with Guide Cone Configuration 1, the cones were place 7.5 feet from the path. Thus the lane remained 15 feet wide. Several cones were placed inside the loops at the intersection to aid in navigation of the straightaway and the s-curve. This new configuration provided the controller with a much better visual to base maneuvers off of. After implementation

of the Guide Cone Configuration 2, the *Sabre* moved through the circuit in a faster, more controlled manner.

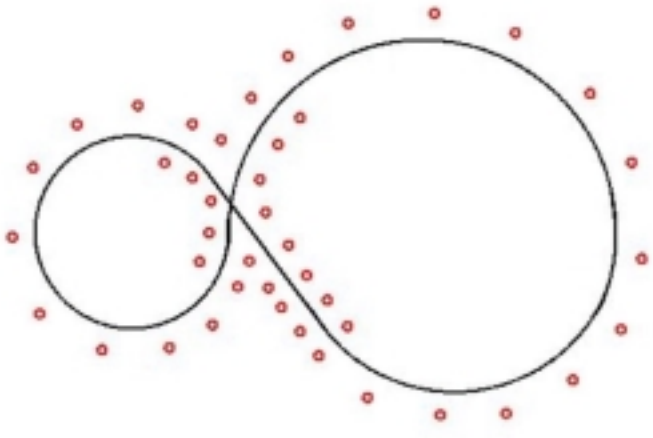

Figure 3. 15: Guide Cone Configuration 2.

3.3 Reconstruction

After testing was concluded, it was necessary to reproduce the data for several reasons. Reproduction of the data into a visual format allows immediate error detection. All programs involving the experimental test data use at least some portion of the reconstruction algorithms. Comparison of reconstruction methods will also be used to for determining trackslip. Visual confirmation of expected vehicle paths is used to add confidence to the reproduction algorithms.

3.3.1 Calculation Assumptions

The reconstruction of the experimental data required assumptions. These assumptions are generated due to the fact that the condition of the testing was not perfect for the acquisition of data. Each of the following assumptions will also discuss conditions that would improve the validity of each assumption.

- *The speed measured by the 5th wheel is accurate.* The 5th wheel had a basic suspension composed of a coil spring and a shock absorber. This simple type of suspension tends to compromise tire contact over rough terrain. This loss of contact over similar terrain increases as speed increases. The assumption being used is that any lack of contact causes a negligible effect to speed. A good check for the validity of this assumption is to look at the data from the high-speed tests. A characteristic of a high-speed test is that the system is driven along a linear path. This minimizes the slip of the track systems and, therefore, allows the track speeds to be a viable comparison with the 5th wheel speed. **Figure 3.16** shows speeds from the *Sabre11* test of 07/15/03 and the *Sabre7* test of 07/16/03. The *Sabre11* high-speed test was over a rougher terrain, whereas the *Sabre7* high-speed test was over a relatively smooth surface. At lower speeds, both the 5th wheel speeds of both tests were very close to the track speeds. It wasn't until the system reached speeds greater than 20 mph on the *Sabre11* test that the speed of the 5th wheel began to fluctuate. Since all of the handling tests maintained speeds less than 20 mph, many of which never even reached 10 mph, this assumption proved reasonable. The best way to decrease contact loss would be to rerun the tests on a smoother terrain. This was not a feasible alternative for July 2003.

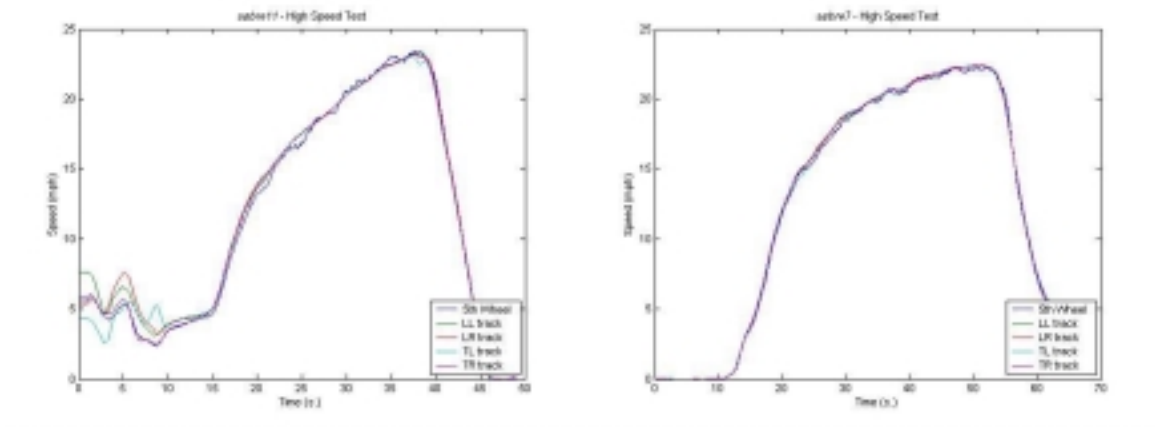

Figure 3. 16: Speed Data for the high-speed tests *Sabre11* (left) and *Sabre7* (right). Compares the track speeds with the 5th wheel speed.

- *The pivot radius for each track system is located at the centerline of each track.* The pivot radius dictates the point of reference for the speed of each track. The centerline of the track systems is located 42.5 inches from the vehicles' centerlines. This value is used to calculate trackslip, **Section 3.3.5**, by relating measured track speeds with theoretical track speeds calculated from yaw rate and 5th wheel speed. Since this is a reference point, there is no need to improve the value. However, it is important that it is noted. Future calculations, such as those discussed in **Section 5.2**, will use this value. It may become necessary to standardize this value.
- *Uniform ground characteristics for each test run.* This assumption was covered in detail in **Section 1.2**. It was necessary to re-note it because this is the chapter that the assumption will be utilized. As with the previous assumption, this assumption primarily affects trackslip calculations and therefore, will not be utilized in the concluded algorithms.

3.3.2 File Structure and Catalog

The Datron system had a file type unique to the system. To help with this, Datron supplied a program that converted all files to files with the extension "*.asc*". This tab-delimited ASCII file type is good, because such files can be easily imported to common programs such as Microsoft Excel and Matlab for processing.

Name	Unit	Description
time	sec.	Time data.
dist	ft.	Vehicle distance from 5th wheel
speed	mph	Vehicle speed from 5th wheel.
l1trk	mph	Left side track speed of the lead vehicle.
r1trk	mph	Right side track speed of the lead vehicle.
l2trk	mph	Left side track speed of the trail vehicle.
r2trk	mph	Right side track speed of the trail vehicle.
tdist	in.	String pot distance of the top, center cylinder.
ldist	in.	String pot distance of the left side cylinder.
press1	psi	Pressure in MP1.
press2	psi	Pressure in MP2.
r1	in.	Right hand brake lever in the lead vehicle.
l1	in.	Left hand brake lever in the lead vehicle.
r2	in.	Right hand brake lever in the trail vehicle.
l2	in.	Left hand brake lever in the trail vehicle.
fwdbk	g	Longitudinal accelerations of the motion pack.
rtlft	g	Lateral accelerations of the motion pack.
updwn	g	Vertical accelerations of the motion pack.
roll	°/s	Roll rate of the motion pack.
pitch	°/s	Pitch rate of the motion pack.
yaw	°/s	Yaw rate of the motion pack.

Table 3. 4: Test data channel list and descriptions.

All of the recorded test data were converted using this program and put into directories representing the respective test date, *030711*, *030715*, and *030716*. Each file has an identical layout of twenty-one columns. Each column represents a Datron channel. All of the columns for each individual test are the same length and are dependant on the length of time of the test that was recoreded. A list of each channel's name, unit, and decription of these channels can be found in **Table 3.4**. The table is set

up so the channels listed first to last are respectively located left to right in the files. The names of the channels are listed in a form compatible with program variables because they will be used in all of the reconstruction programs.

It was foreseen from the beginning of analysis that the data were going to be used in many applications. To make those applications simpler, a catalog program was designed. The original program was called *catalog.m* and was designed to load a specified test data file as a matrix and then slice the matrix into 21 column vectors. These vectors were labeled with their proper name and saved to the workspace for use with other programs. A benefit of using this program was that the other programs could input only the vectors they needed. This would potentially save a good deal of time on some of the larger programs.

The catalog file was modified to add upgrades that would help future calculations even more. The new program, called *modcat.m*, did everything that the original *catalog.m* did, but had extra features. The first addition was a moving average filter that allowed a value, *x*, to be input and each channel would be digitally filtered using the following equation:

$$ch_m = \frac{\sum_{n=0}^{x-1} ch_{m-n}}{x} \tag{3.4}$$

Here, *ch* stand for the channel data and *m* stands for the channel index. This filter would be used on each channel for all indices greater than *x* and would be used to smooth out the data. However, the filter could be bypassed by not entering a value for *x*.

In addition to the filter, correction factors were also included. The yaw rate zero offset was calculated from a specific test run and entered in to the program. Also

included were track over sprocket expansion factors. When the vehicles move, the track shoes push outward on the sprockets. The resultant errors are track speeds measuring slower than they should. These factors were calculated by comparing the 5^{th} wheel speed to the track speeds over a very flat, straight track. The vehicles maintained constant speed to prevent trackslip. Then, the speeds were compared and the correction factors were calculated:

$$c.f. = \frac{speed_{5^{th}\,wheel}}{speed_{tracks}} \tag{3.5}$$

The factors corrected a 9.39% error in track speeds. Theses factors were also included in *modcat.m*.

Lastly, three new outputs were included. The first, called *dt*, is a single value that represents the time step. This value was 0.01 seconds for all of the test data, but this was added so the program could be used for data not covered in this report. A variable labeled *filt_size* was merely the filter size that was input into the program as *x*. The last output is the name of the loaded file minus the extension. This output, labeled *loaded_file*, is used mostly for plotting titles and reports.

3.3.3 Reconstruction Methods

Vehicle path was calculated from speed and yaw rate. Speed was integrated to determine the distance the vehicle traveled. The Matlab command, *cumtrapz*, was used to calculate integrals using the trapezoidal approximation of a definite integral. Given that the speeds of the vehicles did not change very quickly and that the time step was very small, 0.001 sec., the error produced by this method should be very low. Using the

diff command with the distance matrix, another matrix, *Δdiff*, was created that showed the distance the vehicle traveled each time step. The yaw rate was integrated to determine the vehicle bearing, θ. The bearing matrix and the *Δdiff* matrix were used in the following equations to determine time step-difference matrices of Cartesian displacement.

$$\Delta X_m = (\Delta diff_m)\sin\theta_m \tag{3.5}$$

$$\Delta Y_m = (\Delta diff_m)\cos\theta_m \tag{3.6}$$

Using the *cumsum* command, these difference matrices can be summed to give the time dependant Cartesian vectors of the vehicle path.

As mentioned earlier, there are three ways of calculating the vehicle speed and yaw. The following subsections will describe these methods, and the pros and cons of each will be discussed. Equations will also be formulated where necessary.

3.3.3.1 Track Speeds

The tracks are the vehicles' method of locomotion. Without considering trackslip, calculation of speed and yaw rate from the track speeds is a simple matter. **Figure 3.17** can be used as reference for calculation. Vehicle speed is calculated by averaging the two track speeds.

$$speed_{CG} = \frac{speed_{RT} + speed_{LT}}{2} \tag{3.7}$$

The yaw rate can be calculated via the geometric constraint of Eq. (3.8). However, this calculation will produce yaw rate with the units of radians per second, unlike the motion pack yaw rate units of degrees per second. Likewise, reverse calculations must be in the same units.

$$speed_{LT} - speed_{CG} = 42.5in.\left(yaw_{rate}\right) \tag{3.8}$$

$$yaw_{rate} = \frac{speed_{LT} - speed_{CG}}{42.5in.} \tag{3.9}$$

When Eq. (3.7) is substituted in, Eq. (3.9) becomes as follows.

$$yaw_{rate} = \frac{speed_{LT} - speed_{RT}}{85in.} \tag{3.10}$$

However, the speed units used in the equations will be feet per second. Thus, it would be wise to convert the length from inches to feet. This will result in the following final solution.

$$yaw_{rate} = \frac{12\left(speed_{LT} - speed_{RT}\right)}{85\,ft.} \tag{3.11}$$

The problem with using track speeds, is that the tracks slip. This slip causes discrepancies between measured track speeds and those as would be expected from vehicle speed and yaw rate.

$$speed_{\exp} = TS\left(speed_{meas}\right) \tag{3.12}$$

In this case, $speed_{exp}$ is the speed that would be used in Eqs. (3.7) and (3.11). The trackslip coefficient, *TS*, is a function of track speed, terrain properties, and vehicle/system mass properties. **Section 3.3.5** will go into more depth on calculating the *TS* coefficient.

Another problem with using track speed as a method of calculating vehicle speed and orientation is that there are no built-in track speed monitors. Monitors were added for this experiment, but they were attached directly to the outside of the sprocket and would be vulnerable to driving obstacles. Therefore, unless a better way can be

found to accurately determine track speeds, they will remain a questionable source of data in field conditions.

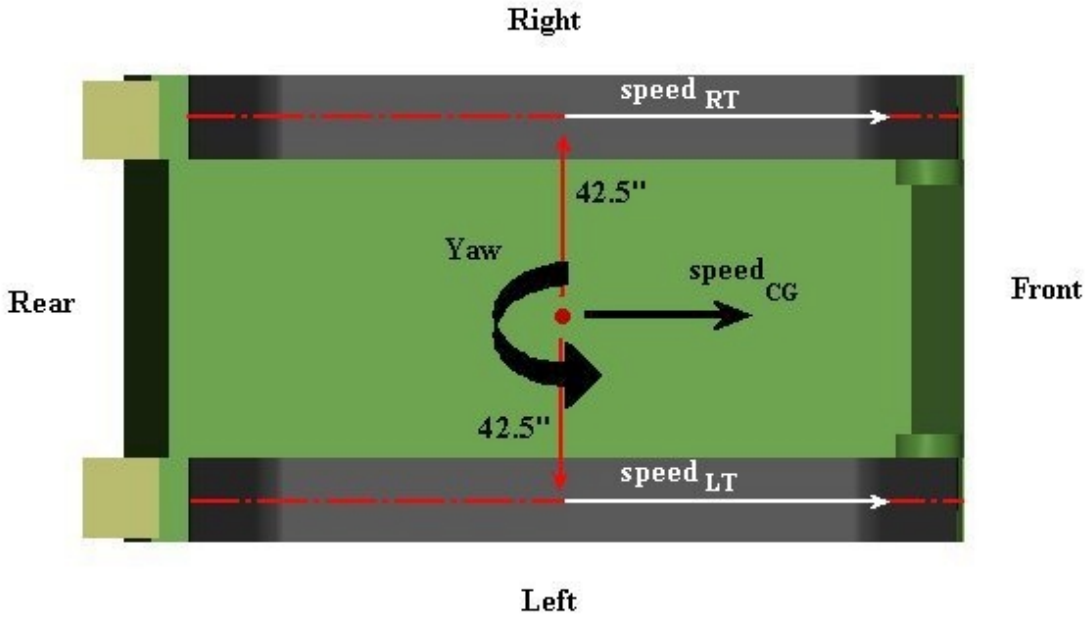

Figure 3. 17: Diagram of M113 used for calculating yaw rate and speed from track speeds.

3.3.3.2 Acceleration and Yaw Rate

Using a motion pack, like the one used in testing, or a similar transducer pack to get accelerations and yaw rate is potentially a compact method for determining vehicle speed and yaw rate. There are several downsides to this method, however. The pack must be located very close to the center of mass of the vehicle. Otherwise, the measurements will be compromised. Speed must be calculated by integrating the accelerometer signal. An initial speed offset must be implemented to make sure the speed calculation is correct. The measurement pack would have to be robust enough to not be damaged by the severe jostling of the system during transit and operation. Minimizing calibrations would be beneficial.

During these tests, there was a calibration error with the linear accelerometers. This corrupted the data, making it unusable. As a result, this method of calculation was not used in the reconstruction of the test data.

3.3.3.3 5th Wheel and Yaw Rate

The main method of calculation for the reconstruction of data for these tests was the 5th wheel speed and the motion pack yaw rate. A nice feature of this method is that there is very little calculation needed and if the 5th wheel is set up properly, there would be no calculation needed. As tested, the 5th wheel was displaced 63 inches off the right side of the trail vehicle. Because of this, the yaw rate affects the measured 5th wheel speed such that it differs from the vehicle speed.

$$speed_{5whl} = speed_{CG} - \frac{63\,ft}{12}\left(yaw_{rate}\right) \tag{3.13}$$

This equation can be rewritten to allow for correction of 5th wheel speed using yaw rate.

$$speed_{CG} = speed_{5whl} + \frac{63\,ft}{12}\left(yaw_{rate}\right) \tag{3.14}$$

In order to remove the need for this correction, the 5th wheel would need to be positioned in line with the vehicles center of mass, preferable beneath the yaw rate transducer. If this were done, the yaw rate and vehicle speed would be at hand with no calculations.

3.3.4 Visualization

During the development of this project, there were several programs that were developed for the purpose of visualization. These programs not only served to visually reproduce the experimental tests of the system, but visually check for calculation errors.

The following paragraphs will detail these visualization programs and, where necessary, figures will be included to assist in the description. All were created in Matlab.

The *allplot.m* program was created for the sole purpose of viewing the data of all channels within a test set. The program creates seven windows. Each window displays data from related channels for easy viewing and comparison. The first window contains two subplots, 5th wheel distance and 5th wheel speed. All other windows contain single plots. The second window contains all five speed measurements, 5th wheel and track speed. See **Figure 3.18** for examples of the first and second windows using a slalom test, *sabre9* from 07/15/03. The third and fourth windows, **Figure 3.19**, contained the TVLA data elements. The third contained the cylinder displacements. These were measurements relative to the location in which they were attached to the cylinders. The fourth window contained the hydraulic pressures in the manifold.

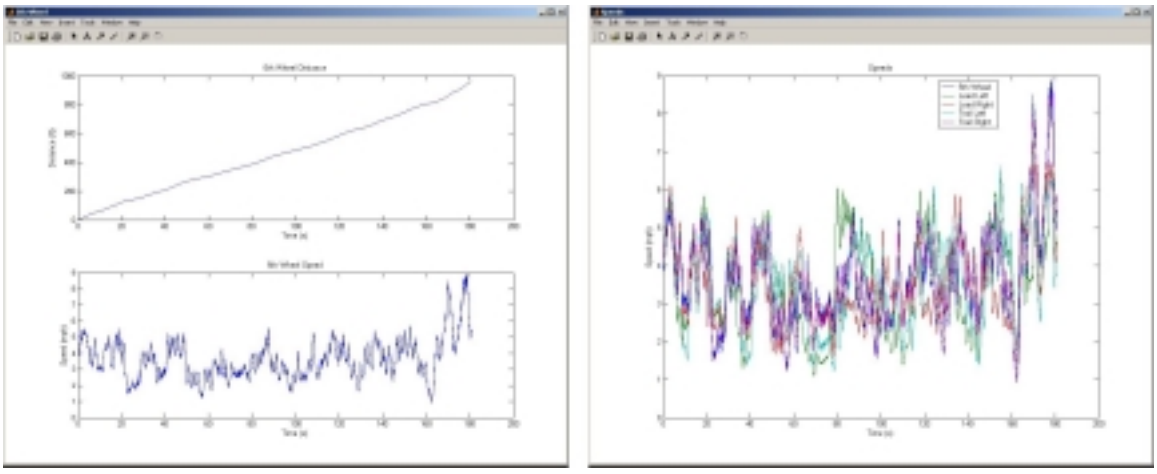

Figure 3. 18: Screen shots of the *allplot.m* windows loaded with *sabre9* data from 07/15/03 and showing 5th wheel displacements and speed (left) and 5th wheel and track speeds (right).

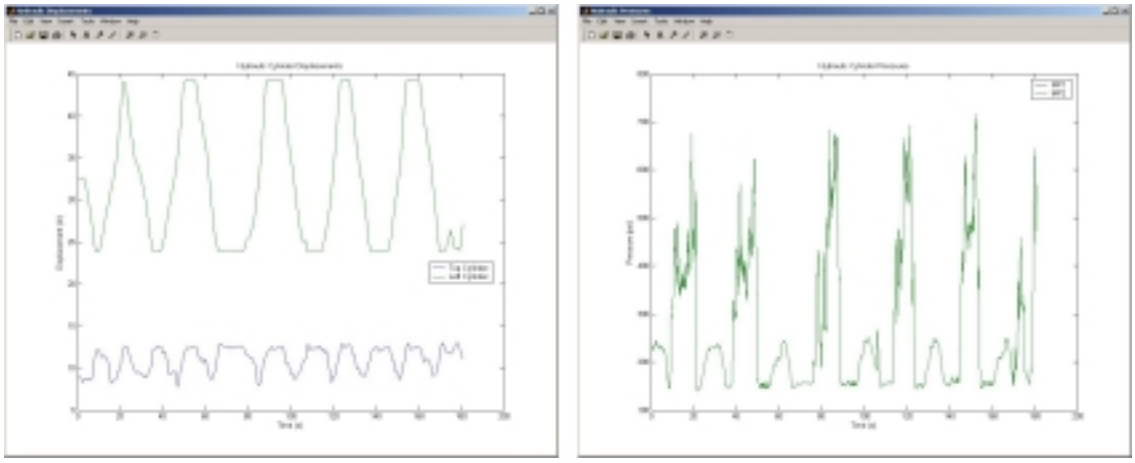

Figure 3. 19: Screen shots of *allplot.m* showing hydraulic cylinder displacements (left) and pressure (right).

The fifth window, **Figure 3.20**, is the plot of the brake stick string pot displacements. **Figure 3.21** shows the sixth and seventh windows. These contain the motion pack measurements. The sixth holds the linear accelerations and the seventh holds the rotational velocities.

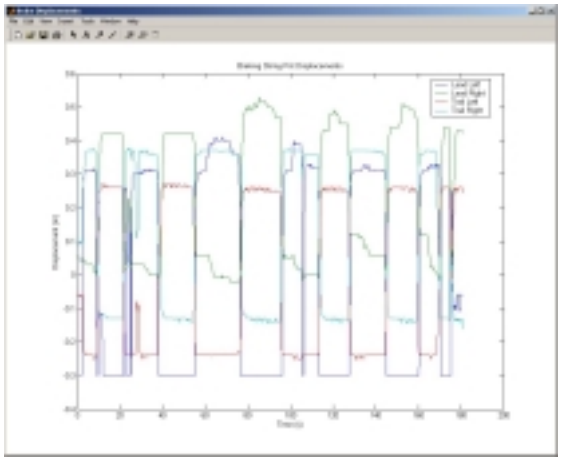

Figure 3. 20: Screen shot of *allplot.m* showing string pot displacements of the brake sticks.

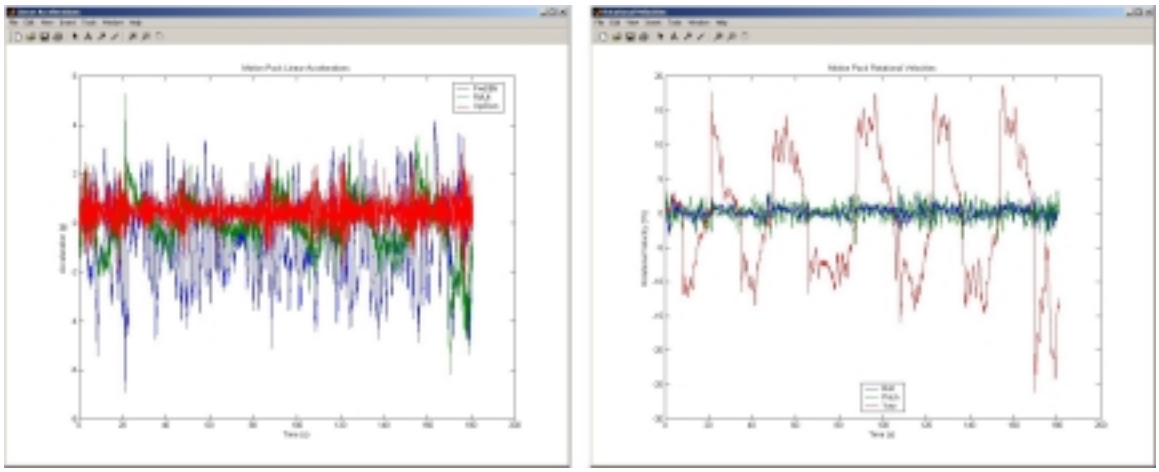

Figure 3. 21: Screen shots of *allplot.m* showing linear acceleration (left) and rotational velocity (right) from the motion pack.

The next programs created were the so-called *mplot* programs. In this case, *mplot* stands for "motion plot" as they were designed to generate partial real-time reconstructions of the data. They are not exactly time coordinated because to do so in Matlab would require more programming and would not give much more to the recreations.

The first of these programs was called *mplot.m*. It used *fwdbk* and *rtlft* accelerations along with *yaw* to calculate the vehicle trajectory. Please refer to **Table 3.4** for descriptions of italicized data. Shortly after the program was created, the flow in the acceleration was made apparent and the program was abandoned. Its successor, *mplot4.m*, calculated the vehicle path using *speed* and *yaw*. Also included, in this program and all future *mplot* programs, was a frame skip option. This allowed the recreations to be viewed at an accelerated frame rate and was particularly helpful when viewing the longer test runs of over five minutes. **Figures 3.22-24** will use the data *sabre4* from 07/16/03.

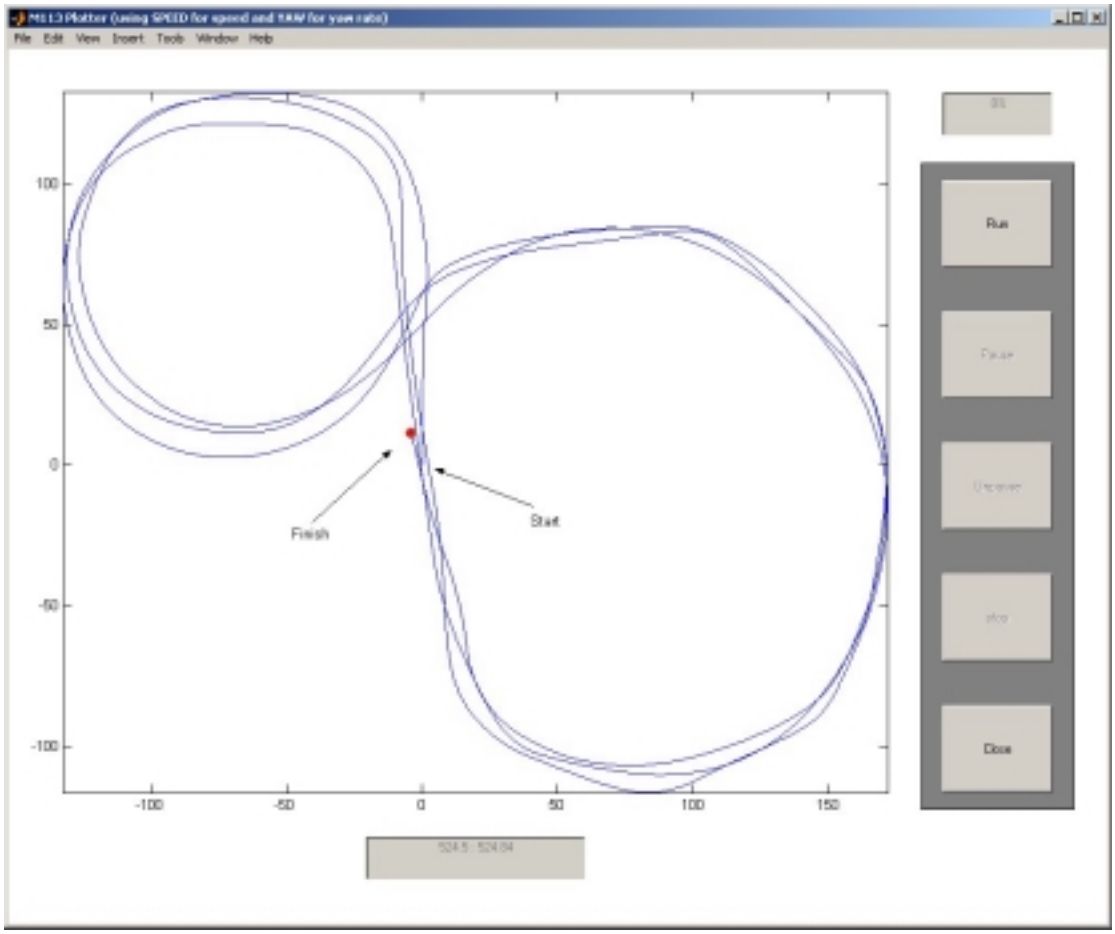

Figure 3. 22: Screen shot of *mplot4.m* using *speed* and *yaw* to calculate vehicle trajectory.

This program was modified to also plot a vehicle path as calculated from the track speeds, *r2trk* and *l2trk*. Renamed *mplot4b.m*, the new program made no trackslip calculations (*ts*=1, see **Section 3.3.5**), but did allow for adjustment of the pivot radius which is 42.5 inches in **Figure 3.23** Therefore, *mplot4b.m* give a good overview as to the effect trackslip has on vehicle track systems.

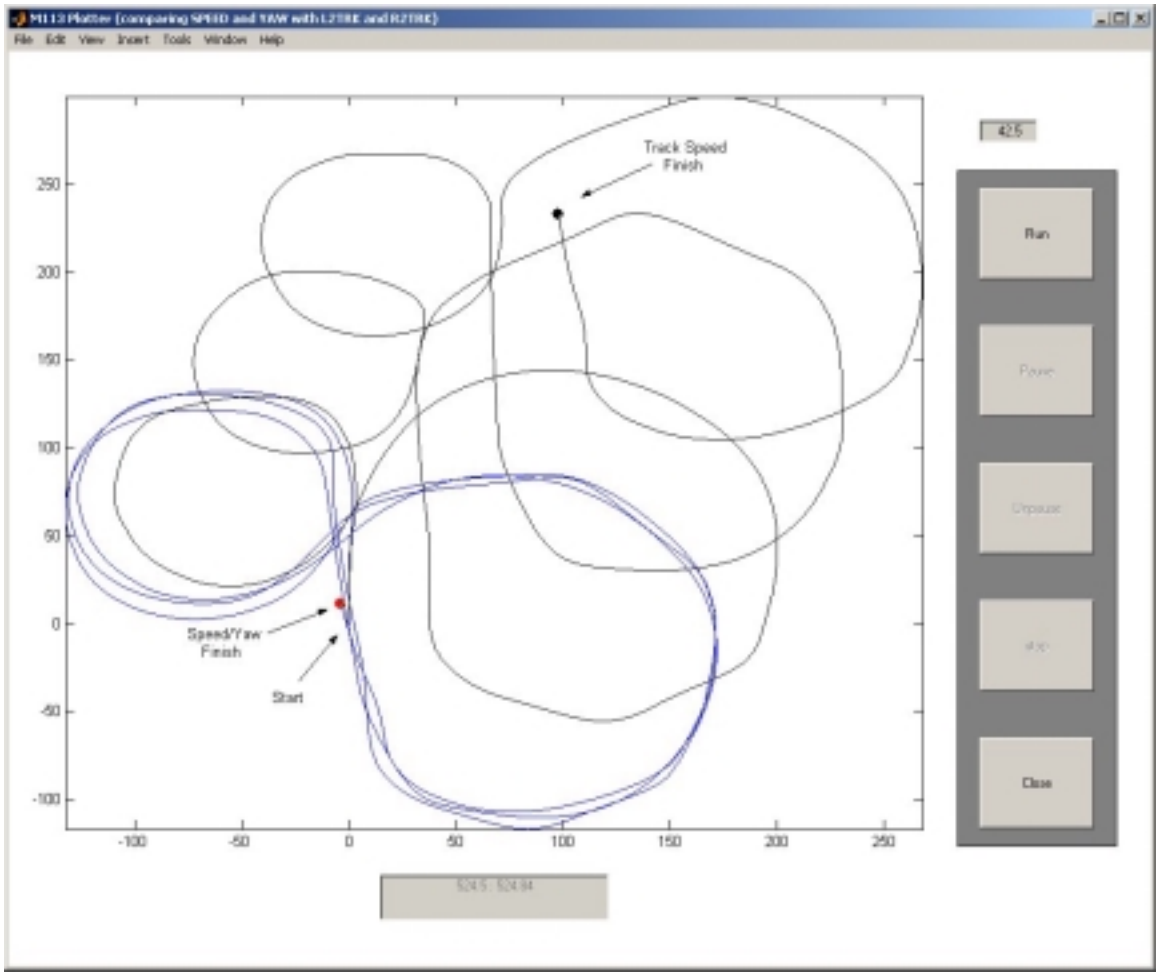

Figure 3. 23: Screen shot of *mplot4b.m* showing vehicle paths calculated from *speed* and *yaw* (blue) as well as unadjusted track speeds, *r2trk* and *l2trk* (black).

The *mplot4.m* program was further developed to include both vehicles. The *ldist* channel was used to calculate the γ angle between the TVLA and the trail vehicle. This version was named *mplot5.m*. Boxes were generated to represent the two M113s and a triangle was created as the TVLA. From this, two more similar programs were created. The program *mplot6.m* was developed to record an AVI video file of the same animation generated by *mplot5.m*. In *mplot7.m*, data were generated showing the area the system covered during the run. Trails were drawn from outside points on each vehicles hull directly out sideways from the vehicles' centers of mass. This was intended to be a step

toward a study in mine path coverage, but turned out to be of little benefit due to the fact that the system was not installed with mine rollers.

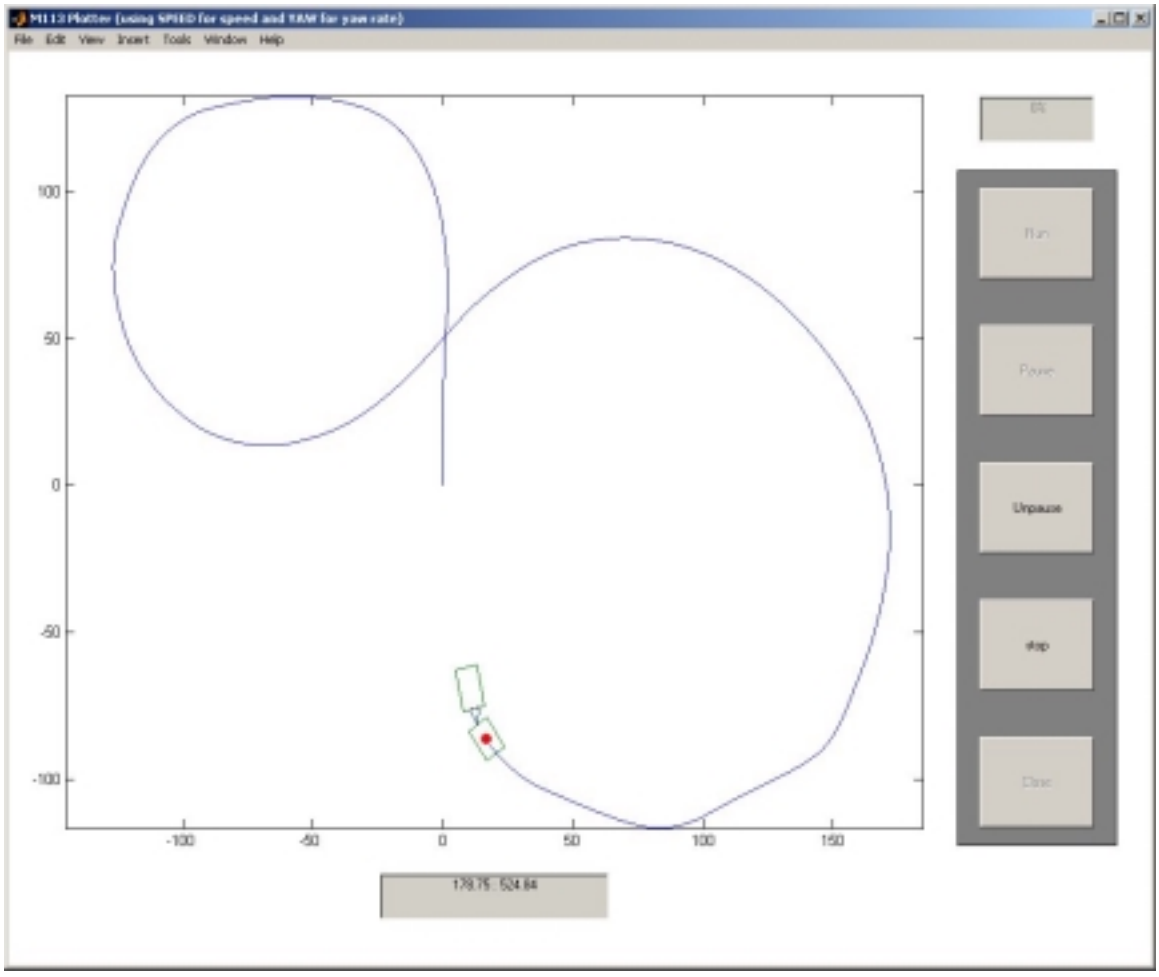

Figure 3. 24: Screen shot of *mplot5.m* in mid-run (178.75 sec. of 524.84 sec.).

3.3.5 Trackslip

The vehicles are controlled by adjusting track speeds. It is obvious that trackslip will play a critical part in the implementation of the algorithms developed in this project. It was this understanding that started the development of the trackslip calculation program, *trkslpcalc.m*.

The trackslip coefficient, *TS*, is calculated by taking an accepted value of speed and yaw, in this case, using the 5th wheel speed and motion pack yaw, and reverse calculating an expected set of track speeds that would theoretically produce that speed and yaw. Using Eq. (3.8) and (3.10):

$$speed_{LT(EXP)} = speed_{CG} + 42.5in.(yaw_{rate}) \tag{3.15}$$

$$speed_{RT(EXP)} = speed_{CG} - 42.5in.(yaw_{rate}) \tag{3.16}$$

The trackslip coefficient for each track can be calculated by using the following function as calculated from Eq. (3.12):

$$TS = \frac{speed_{EXP}}{speed_{MEASURED}} \tag{3.17}$$

During its development *trkslpcalc.m* became more than just a simple calculation program. Multiple new features were added and the complexity of the program began. Making use of a graphical user interface (GUI), *trkslpcalc.m* is an interactive display program (see **Figure 3.25**).

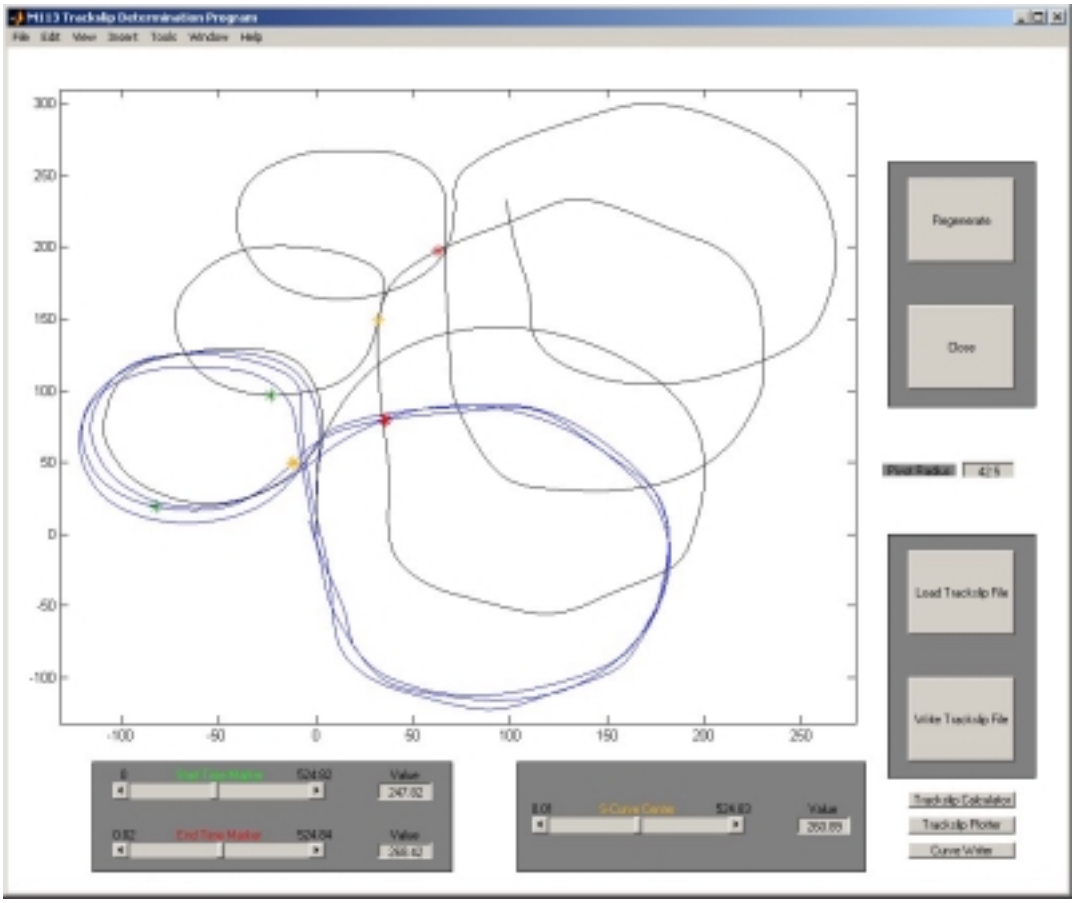

Figure 3. 25: Screen shot of *trkslpcalc.m* after initial startup showing vehicle paths using *speed* and *yaw* (blue) and track speeds, *r2trk* and *l2trk* (black).

As in the *mplot4b.m* program, *trkslpcalc.m* also implements a user-variable pivot radius. The paths need to be manually redrawn after adjusting the pivot radius by clicking the "Regenerate" button. The top button on the lower right corner is a trackslip calculation button. Using the equations found in **Section 3.3.3.1**, the trackslip variables, *TSL* and *TSR*, are found for every time step. When the track speeds reach zero, the *TS* value approaches infinity. In these cases, the calculator rounds to a lower value. These values are then saved to a file of the user's choice. The "Load Trackslip File" button can, then, be used to load the file. Regenerating the data will show a corrected track speed curve.

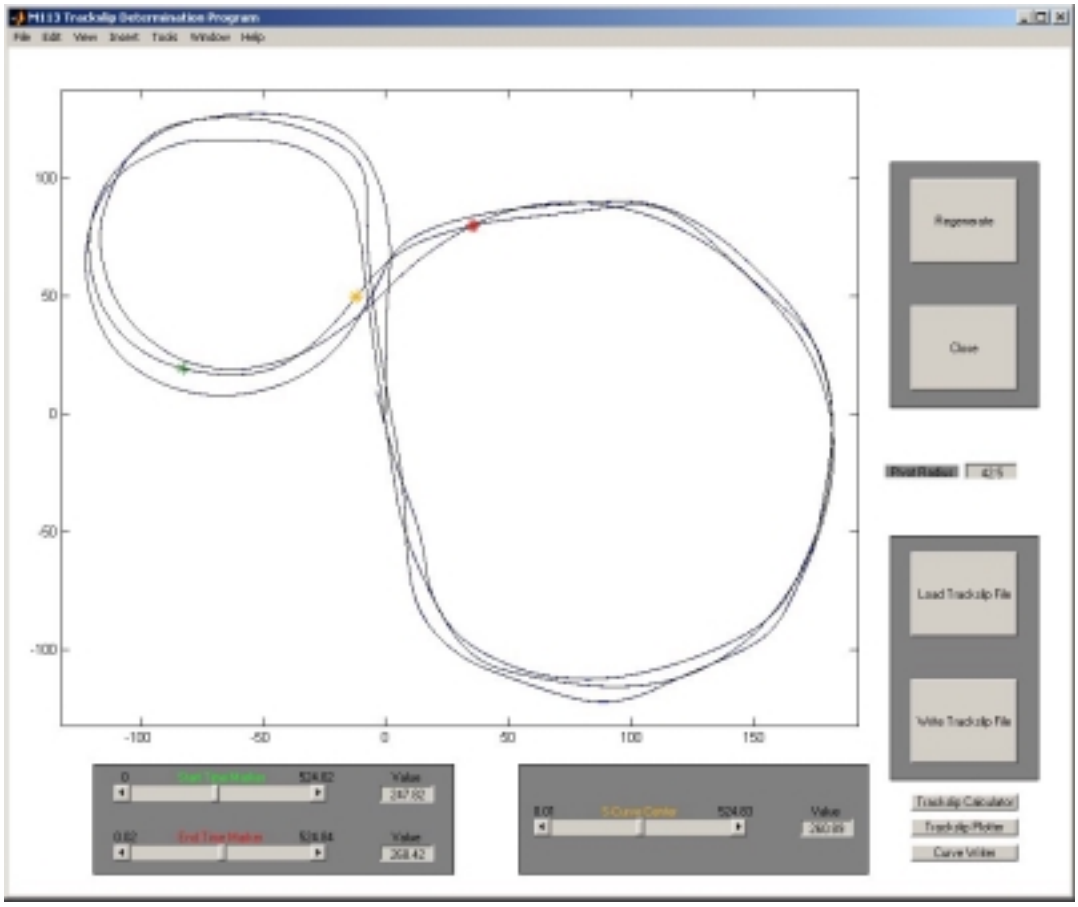

Figure 3. 26: Screen shot of *trkslpcalc.m* after trackslip calculation and implementation.

At any point, the "Trackslip Plotter" button can be clicked to show the trackslip coefficient values that are currently loaded. All trackslip coefficients are initialized to 1.0 at startup. The values can also be saved to a file using the "Write Trackslip File" button.

One of the most important functions of the *trkslpcalc.m* program is the dissection function. Sections of the test can be saved independently of the rest. The sliders below the main display can be used to move the cursors on the paths. The green asterisk marks the beginning of the segment. The red asterisk marks the end of the segment and the orange asterisk marks the midpoint of an s-curve. This asterisk is irrelevant if the curve is not an s-curve. When the desired segment is selected, the "Curve Writer" button can be clicked to run *crvwrite.m*, bringing up its GUI.

Figure 3. 27: Screen shot of *crvwrite.m* GUI with "Curve Type" pull down menu exposed.

After entering a proper title and description, the type of curve must be selected. The curve type selected will only affect the information file generated by *crvwrite.m*. Then the curve can be written to disk. When a file name is given, the program makes two files. Under that file name is all of the raw data for that time segment. The file shares the same name with a prefix of "info_". This file includes the title and description as well as the variables included in the other file. Also included in the info file is curve specific data such as straightaway length, curve length, curve angle, curve radius, and/or average turn rate. If the curve type was an s-curve, there would be data on both halves of the s-curve, as defined by the orange asterisk from *trkslpcalc.m*. The info file can be easily read into Microsoft Excel for review.

As will be discussed in the **Section 5.2**, use of the trackslip function, Eq. (3.12), and computation of the trackslip coefficient, *TS*, will be needed. It was concluded that, even though *TS* could be computed, the data recorded was unacceptable for use in generalizing *TS*. Most of the measurements from 07/15/03 were conducted on the Spin-Up Packed Snow Area. Half of the area has a packed-down broken asphalt surface. The other half has a softer, grass-covered soil. The 07/16/03 measurements were conducted on the Spin-Up Packed Snow Area, the gravel access roads, and the "ice rink". There

was too much variation in ground type between tests to use the data for the characterization of trackslip properties.

To show the variation in trackslip, a trackslip program was created. The final version of this program, *ts_least_sq2.m*, loaded all of the left or right turns, listed in **Table 5.1**, and calculated the average trackslip for each track. This version of the program also calculates a best-fit surface using all of the average values. The best-fit equation used is:

$$f(v,r) = a_1 v \ln(r-m) + a_2 \ln(r-m) + a_3 v + a_4 \approx TS \qquad (5.1)$$

In this equation, *v* stands for average velocity and *r* stand for average radius of curvature. The variable, *m*, represents the expected minimum turning radius as observed by looking at trackslip as function of radius. The *m* variable was 22.4170 ft. for left turns and 25.7639 ft. for right turns.

Figures 5.1 – 5.4 are the results of the trackslip program. The three-dimensional plot in the upper right corner of each figure shows the data points relative to the best-fit surface calculated from Eq. (5.1). The other two plots are the mean squared error between the data and the best-fit equivalents. The plots on the left show the data as a function of radius of curvature and the right plots show the data as a function of average velocity.

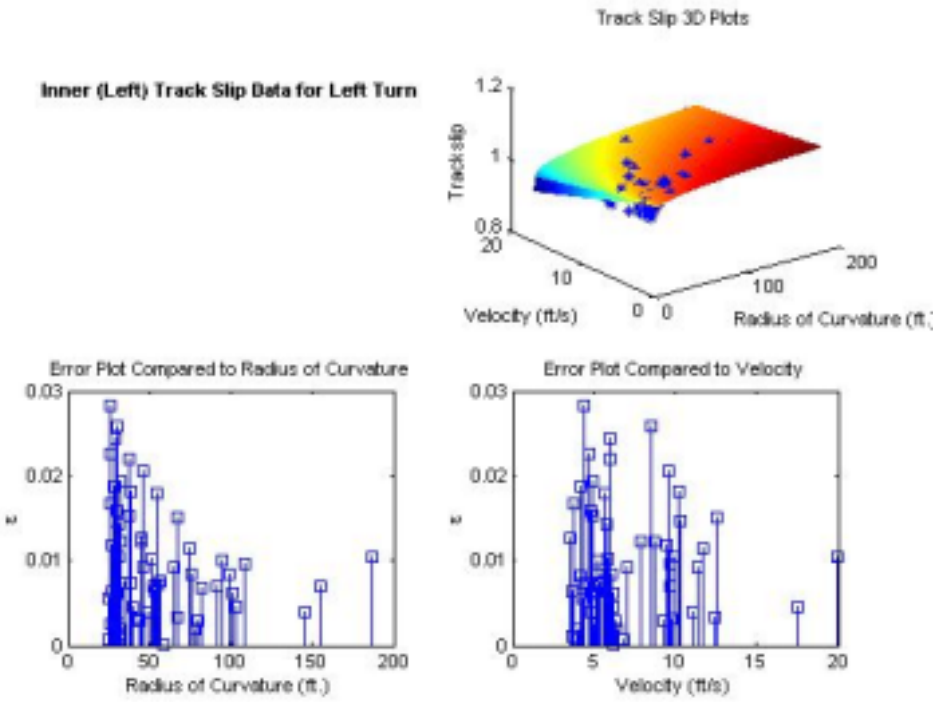

Figure 3. 28: Error in trackslip and best-fit surface for left tracks during left turns.

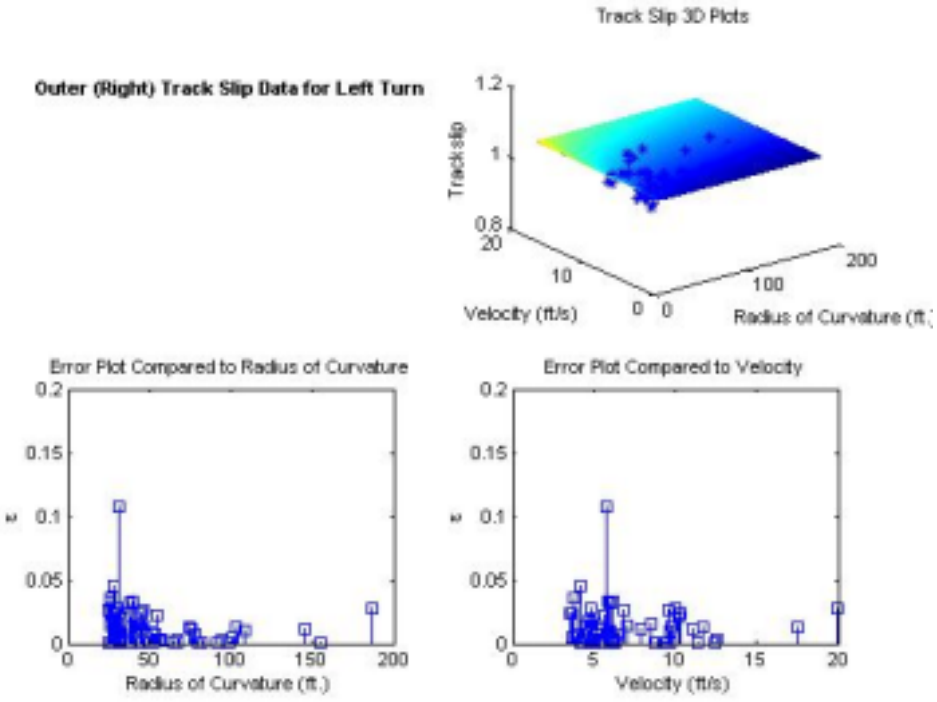

Figure 3. 29: Error in trackslip and best-fit surface for right tracks during left turns.

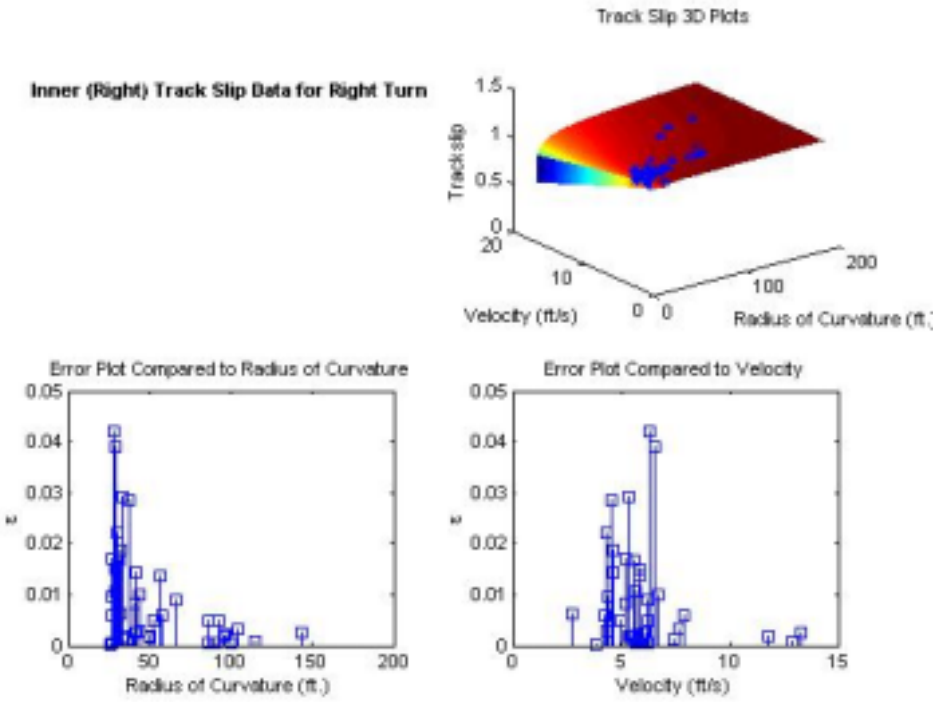

Figure 3. 30: Error in trackslip and best-fit surface for right tracks during right turns.

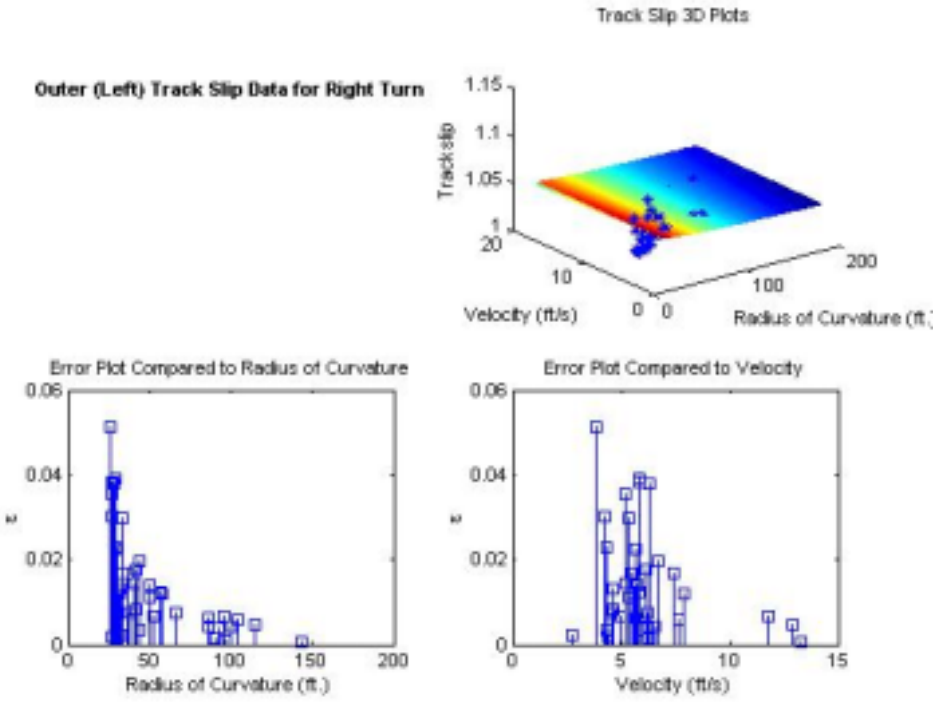

Figure 3. 31: Error in trackslip and best-fit surface for left tracks during right turns.

The error of the best-fit surface decreases as the radius of curvature increases as can be seen in the previous pictures. There is too much variation to make any conclusions on trackslip values. Further steps in determining trackslip will be recommended in **Section 5.2**.

Chapter 4. Theory and Correlation

4.1 Model Generation

4.1.1 Theory

As stated in **Section 1.**1, this project is the first step in order to automate the trail vehicle in the *Panther Lite* system and that first step is the generation of algorithms that control the motion of the trail vehicle. The algorithms will use the state of the lead vehicle and the TVLA in order to calculate the necessary trail vehicle response. The algorithms were generated with the purpose of reducing stress to the TVLA.

Keeping the idea of reducing stress in mind, the trail vehicle was considered to be a trailer. This means that the system will be looked at as if the lead vehicle were towing the trail vehicle. Obviously, if this was truly happening, there would be larger stresses in the TVLA. However, this motion is natural, and if the trail vehicle could be made to imitate the trailer motion, there would be very little stress in the system. The key to this is the connection point between the TVLA and the trail vehicle. With the assumption of two-dimensional motion as described in **Section 1.2**, the lead vehicle and the TVLA form a single rigid body. Therefore, the motion of the lead vehicle defines the trajectory of this connection point.

4.1.2 Calculation

Plane kinematics states that any point on a rigid body has a calculable instantaneous velocity. The velocity of a specific point is a vector sum of the velocity of the body's center of mass and the velocity of the point relative to the center of mass.

This relative velocity can be measured as the cross product of the body's rotation about that center of mass and the position vector of the measured point to the center of mass (ex. Eq. (4.1)). The connection point between the TVLA and the trail vehicle is part of two rigid bodies. They are the lead vehicle/TVLA and the trail vehicle. See **Figure 4.1** for reference.

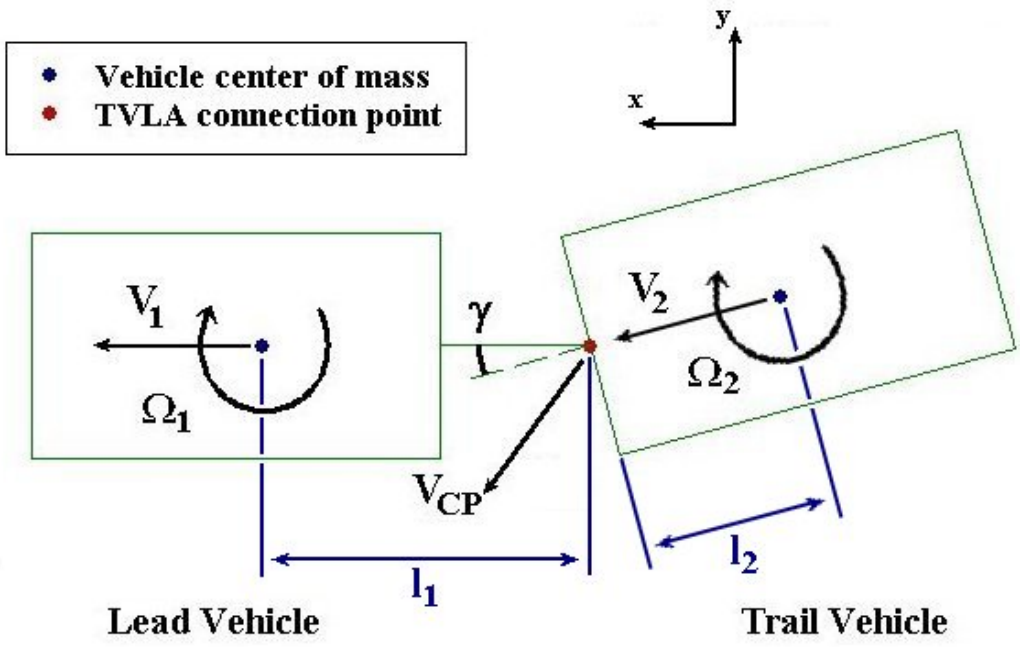

Figure 4. 1: Kinematic model of the setup as tested.

Because the connection point is part of two rigid bodies, two equations can be set up. The first calculating the connection point velocity, $\overline{V}_{CP}$, from the lead vehicle. The second will calculate $\overline{V}_{CP}$ from the trail vehicle.

$$\overline{V}_{CP} = \overline{V}_1 + \overline{V}_{CP/1} \tag{4.1}$$

$$\overline{V}_{CP} = \overline{V}_2 + \overline{V}_{CP/2} \tag{4.2}$$

Substituting Eq. (4.1) into Eq. (4.2) forms:

$$\overline{V}_1 + \overline{V}_{CP/1} = \overline{V}_2 + \overline{V}_{CP/2} \tag{4.3}$$

In component form, Eq. (4.3) becomes:

$$\begin{Bmatrix} v_1 \\ 0 \end{Bmatrix} + \begin{Bmatrix} 0 \\ -\Omega_1 l_1 \end{Bmatrix} = \begin{Bmatrix} v_2 \cos\gamma \\ -v_2 \sin\gamma \end{Bmatrix} + \begin{Bmatrix} l_2\Omega_2 \sin\gamma \\ l_2\Omega_2 \cos\gamma \end{Bmatrix} \tag{4.4}$$

In this equation ν_1 and ν_2 are scalar magnitudes of the velocity vectors, V_1 and V_2, respectively. Viewing the equation in this form allows orthogonal parts of the vector to be split into separate equations.

$$v_1 = v_2 \cos\gamma + l_2\Omega_2 \sin\gamma \tag{4.5}$$

$$-\Omega_1 l_1 = -v_2 \sin\gamma + l_2\Omega_2 \cos\gamma \tag{4.6}$$

The next series of equations will show the progression of using Eqs. (4.5) and (4.6) to calculate the functions for $\nu_2(\gamma,\nu_1,\Omega_1)$ and $\Omega_2(\gamma,\nu_1,\Omega_1)$. From Eq. (4.5):

$$v_2 = \frac{v_1}{\cos\gamma} - \frac{l_2\Omega_2 \sin\gamma}{\cos\gamma} \tag{4.7}$$

$$v_2 = v_1 \sec\gamma - l_2\Omega_2 \tan\gamma \tag{4.8}$$

Substituting Eq. (4.8) into Eq. (4.6):

$$-\Omega_1 l_1 = -v_1 \tan\gamma + l_2\Omega_2 \sin\gamma \tan\gamma + l_2\Omega_2 \cos\gamma \tag{4.9}$$

$$\Omega_2 \left(l_2 \sin\gamma \tan\gamma + l_2 \cos\gamma \right) = v_1 \tan\gamma - \Omega_1 l_1 \tag{4.10}$$

$$\Omega_2 = \frac{v_1 \tan\gamma - \Omega_1 l_1}{l_2 \sin\gamma \tan\gamma + l_2 \cos\gamma} \tag{4.11}$$

Eqs. (4.11) is the desired function. However, it is a fairly complex equation. Reduced equations improve calculation time. This is beneficial when algorithms are implemented and need to make real-time calculations. Eq. (4.11) can be reduced using basic trigonometric identities. The following equations show this process.

$$\Omega_2 = \frac{v_1 \frac{\sin\gamma}{\cos\gamma} - \Omega_1 l_1}{l_2 \frac{\sin^2\gamma}{\cos\gamma} + l_2 \cos\gamma} \tag{4.12}$$

$$\Omega_2 = \frac{v_1 \frac{\sin\gamma}{\cos\gamma} - \Omega_1 l_1}{l_2\left(\frac{\sin^2\gamma}{\cos\gamma} + \frac{\cos^2\gamma}{\cos\gamma}\right)} \tag{4.13}$$

$$\Omega_2 = \frac{v_1 \frac{\sin\gamma}{\cos\gamma} - \Omega_1 l_1}{l_2}\cos\gamma \tag{4.14}$$

$$\Omega_2 = \frac{v_1 \sin\gamma - \Omega_1 l_1 \cos\gamma}{l_2} \tag{4.15}$$

Trail vehicle speed can then be found by substituting Eq. (4.15) into Eq. (4.8):

$$v_2 = v_1 \sec\gamma - l_2 \tan\gamma\left(\frac{v_1 \sin\gamma - \Omega_1 l_1 \cos\gamma}{l_2}\right) \tag{4.16}$$

This equation can be reduced as follows.

$$v_2 = v_1 \frac{1}{\cos\gamma} - v_1 \frac{\sin^2\gamma}{\cos\gamma} + \Omega_1 l_1 \sin\gamma \tag{4.17}$$

$$v_2 = v_1\left(\frac{1}{\cos\gamma} - \frac{\sin^2\gamma}{\cos\gamma}\right) + \Omega_1 l_1 \sin\gamma \tag{4.18}$$

$$v_2 = v_1 \cos\gamma + \Omega_1 l_1 \sin\gamma \tag{4.19}$$

Given the speed of the center of mass of the lead vehicle, the rotational velocity of the lead vehicle, and the angle between the lead vehicle and the trail vehicle, Eqs. (4.15) and (4.19) calculate the speed and rotation of the trail vehicle. The equations were reduced to sums of basic trigonometric functions, all of gamma, with coefficients. These calculations require very little calculation time. The units for speed, length, and rotational velocity are feet per second, feet, and radians per second, respectively. However, this equation will still work using meters per second and meters as the units for speed and length. All other units would have to be converted before use in the equations.

4.1.3 Simulation

With the algorithms calculated, simulations were conducted. These were prepared because they could show what should be close to the expected results from the experiments. However, the experimental results were expected to be different because the lead vehicle was constantly changing speed and rotational velocity. In case of the simulations, the lead vehicle speed was maintained at a constant rate of 15 feet per second and the rotational velocities remained constant over discrete curves and zero when moving down a straightaway.

The simulations were done using Matlab. A program called *pscalc.m* was created as a template for all the simulations. Within this program were the calculations that took vectors of the lead vehicle speed and rotational velocity and using the algorithms, Eqs. (6.15) and (6.19), produced the speed and rotational velocity of the trail vehicle. The template file was used to create three unique programs for simulation of different vehicle maneuvers. All of these files are identical with two exceptions. Toward the beginning of

each file, the lead vehicle rotational velocity vector is designated. Because of the simple nature of the simulations, these vectors tended to be step functions in order to maintain the desired vehicle path. There are also labeling variations in the plots that assist in distinguishing between the maneuvers simulated.

The first simulation program was called *pscalc1.m*. It simulated the system traveling around a straight-sided oval track. The sides of the track were 90 feet long. The semicircular ends had a 60-foot radius. The lead vehicle started 30 feet before the first turn to the right. The vehicle then proceeded around the track, past the starting point to stop at the beginning of the turn. With the lead vehicle travelling at the designated 15 feet per second, the simulated time was 39.13 seconds.

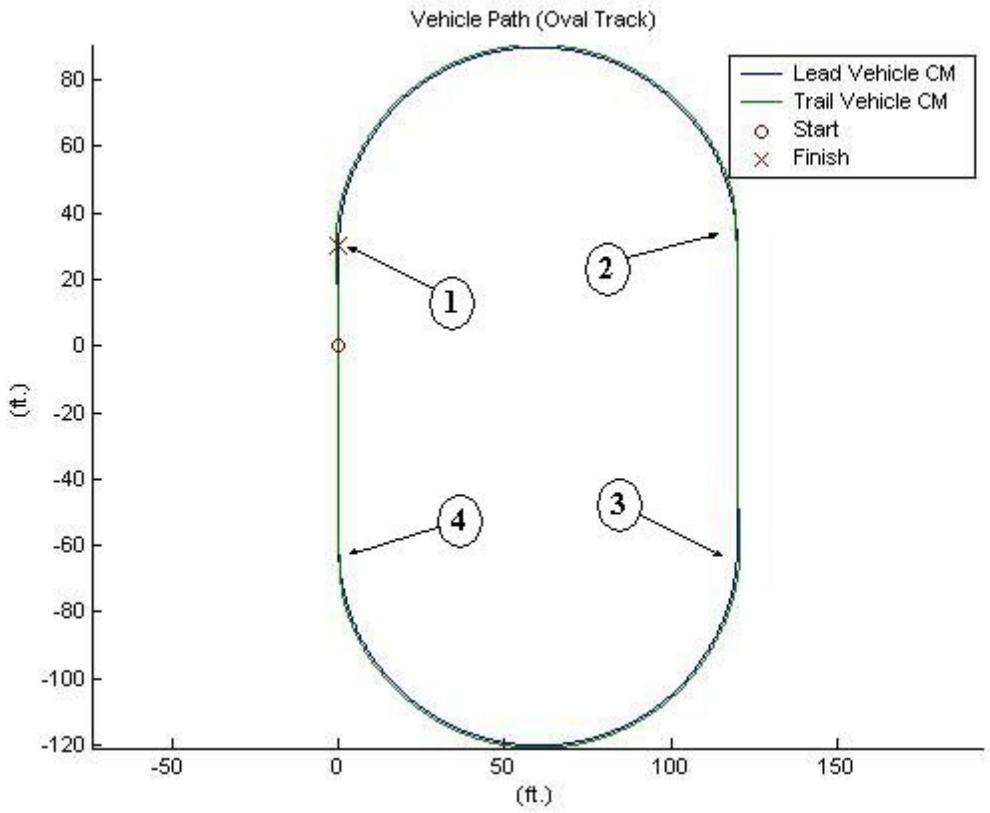

Figure 4. 2: Vehicle path results for the Oval Track simulation.

The numbered indexing in **Figure 4.2** and repeated in **Figure 4.3** marks changes in rotational velocity steps. They are described in detail in **Table 4.1**.

Index	Description
1	Beginning of first 60-foot radius semicircular turn.
2	Beginning of first full 90-foot straightaway.
3	Beginning of second 60-foot radius semicircular turn.
4	Beginning of second full 90-foot straightaway.

Table 4. 1: Numerical index descriptions for Oval Track simulation results.

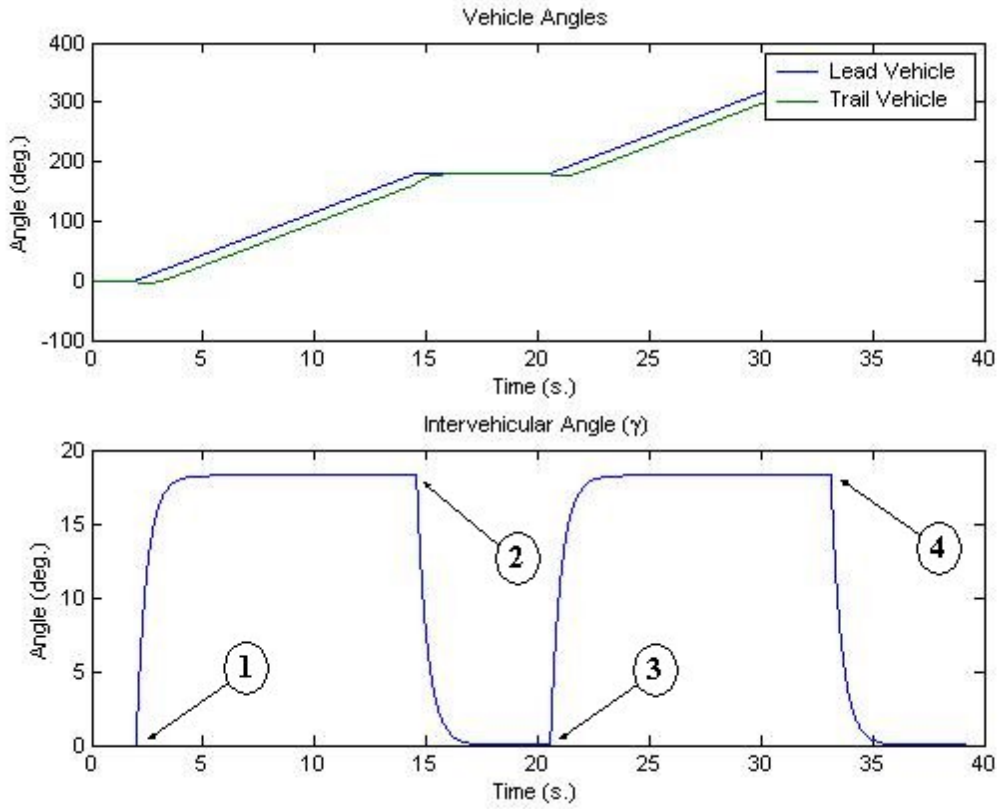

Figure 4. 3: Plots of vehicle angles from Oval Track simulation results.

These results show that when the lead vehicle began a turn, it would take about a second for the trail vehicle to match the lead vehicle's rotational velocity. At this point, the inter-vehicular angle, γ, would reach a steady state value of approximately 18

degrees. It took approximately the same amount of time for the system to straighten out for the straightaway.

The program, *pscalc2.m*, simulated a slalom test similar to the *sabre8*, *sabre9*, and *sabre10* tests on 07/15/03. The simulated system made a 30-foot straight approach at a 30-degree angle to the virtual line of cones. Then the system began a series of 60-degree arcs of 50-foot radius beginning with a left turn, then a right and so on. The system slalomed around nine virtual cones, each spaced 50 feet apart. The simulation recorded 64.82 second of run time.

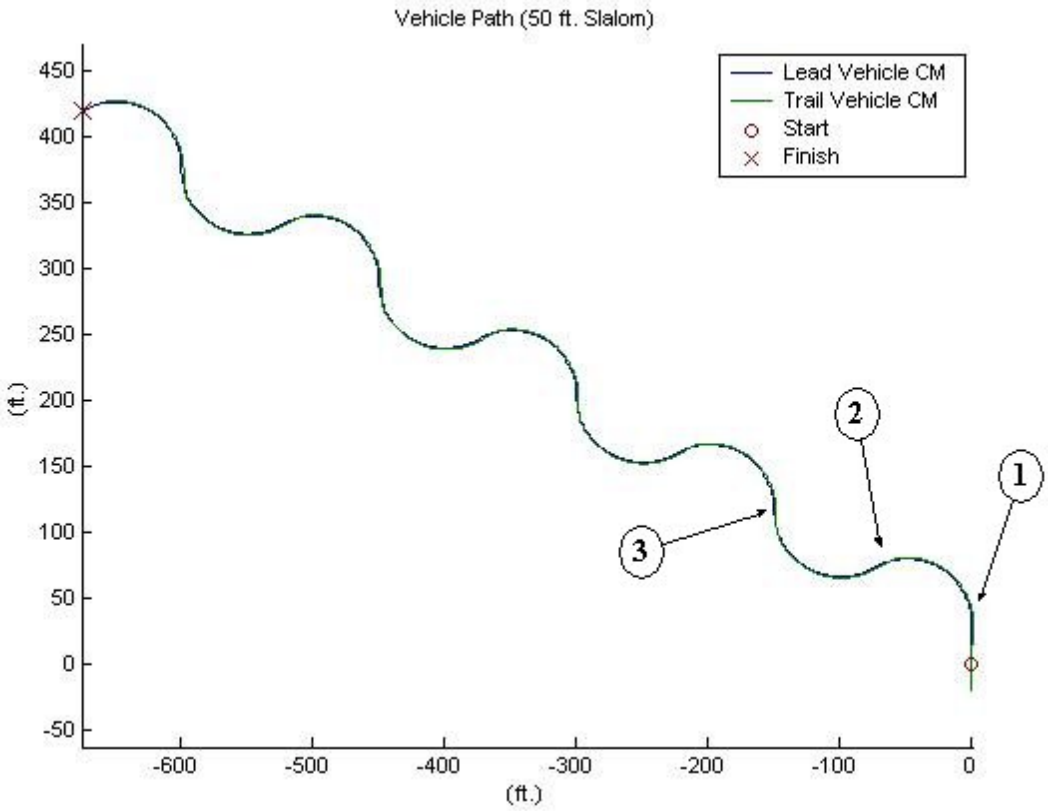

Figure 4. 4: Vehicle path results for the Slalom simulation.

As with the oval track test, **Table 4.2** includes the description of the numerical indexing as found in **Figure 4.4** and **Figure 4.5**. Indices 2 and 3 are repeated three more times throughout the run, but there was no need to repeat the labeling.

Index	Description
1	Beginning of first 60-foot radius left turn from approach.
2	Beginning of 60-foot radius right turn.
3	Beginning of 60-foot radius left turn.

Table 4. 2: Numerical index descriptions for Slalom simulation results.

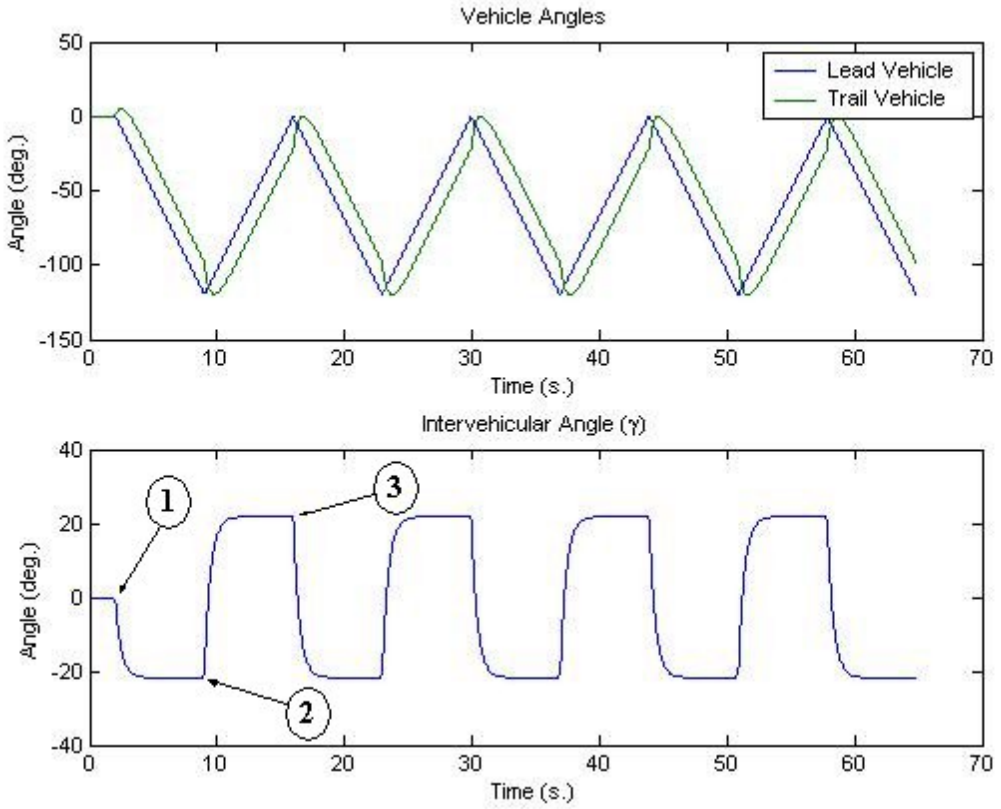

Figure 4. 5: Plots of vehicle angles from Slalom simulation results.

The behavior of the system in this simulation looks similar to the results from the oval track simulation. It is noticeable, however, that the steady state γ angles are around ±21 degrees unlike the 18 degrees from the previous simulation. This is due to the tighter

turns, 50-foot radius instead of 60-foot radius. The greater the turn radius, the closer to zero the steady state γ angle will become.

The last program was called *pscalc3.m*. This program simulated the modified figure eight test that the system was navigated through for the *sabre2*, *sabre3*, and *sabre4* tests on 07/16/03. Virtually, the system began at the point where the straightaway connected to the 75-foot radius curve. The system followed the straightaway and made a tight left turn around a 45-foot radius arc. After approximately 300 degrees of rotation, the lead vehicle began a 90-foot radius right turn that lasted for 180 degrees. At this point, the 90-foot radius right turn became a 75-foot radius right turn. This arc continued until it reached the beginning of the straightaway. At this point, it retraces its path for the length of the straightaway. With the designated speed, the system ran for a simulated time of 60.25 seconds.

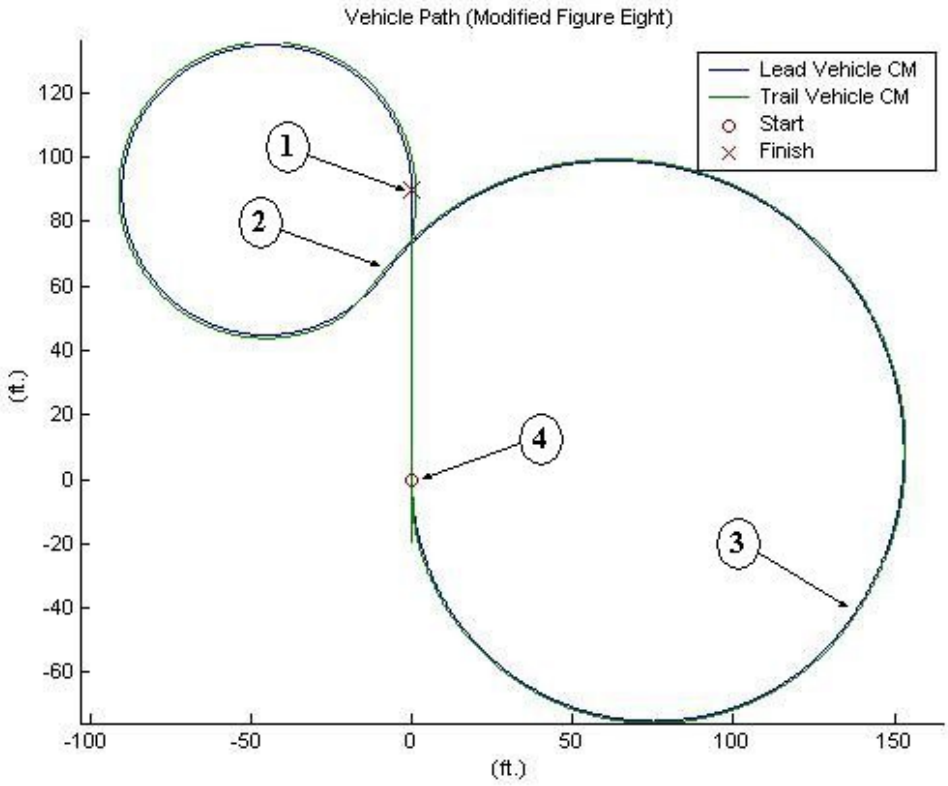

Figure 4. 6: Vehicle path results for the Modified Figure Eight simulation.

The results were predicted to be similar to the previous two simulation results. The steady state γ angle is expected to have a magnitude greater that 21 degrees on the 45-foot radius turn. The value should be negative because it is a left turn. The angle should be less than 18 degrees on the 75-foot radius curve and even lower than that for the 90-foot radius curves. Again, the descriptions for the numerical indexing in **Figure 4.6** and **Figure 4.7** can be found in **Table 4.3**.

Index	Description
1	Beginning of 45-foot radius left turn.
2	Beginning of 90-foot radius right turn.
3	Beginning of 75-foot radius right turn.
4	Beginning of straightaway.

Table 4. 3: Numerical index descriptions for Modified Figure Eight simulation results.

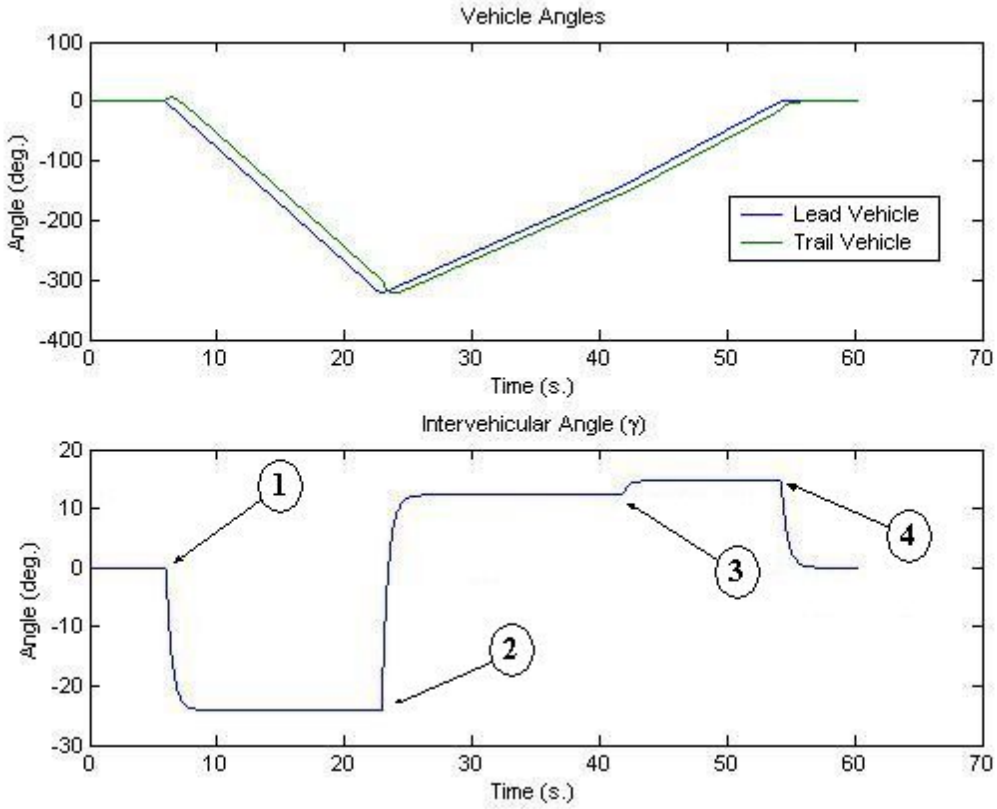

Figure 4. 7: Plots of vehicle angles from Modified Figure Eight simulation results.

The results were as expected. The γ angle reached a steady state value of around -24 degrees on the 45-foot radius curve, around 14 degrees on the 90-foot radius curve, and around 16 degrees on the 75-foot radius curves. Since the simulation results came back as expected and appear acceptable, it was time to implement the algorithms into the experimental data and correlate the theoretical results with the experimental results.

4.2 Correlation

Theory is validated by correlating calculated values with experimental data. In the case of this project, the theoretical data is supposed to be close to the experimental data, but not exactly. This is because the theoretical data represents the optimal reactions

of the trail vehicle. During the experiments, the trail vehicle was controlled independently from the lead vehicle. It was not known, during these tests, how to maintain optimal operation of the trail vehicle.

Matlab was used to create the correlation program. This program, called *corrcalc.m* uses data from the tests described in **Section 3.2.4.4** and **Section 3.2.4.5**. The catalog program, *modcat.m*, was utilized to load the test data before *corrcalc.m* was used. The correlation program produces two output graphs per test. The first plot shows three center of mass paths. The first belongs to the lead vehicle. The second belongs to the trail vehicle as from the experimental data. The third path is from the trail vehicle as calculated by the theoretical algorithms. This path is labeled “Opt. TV” which stands for “optimal trail vehicle”. The second graph is a set of time plots of the inter-vehicular angle, γ, both experimental theoretical (again labeled as “optimal”).

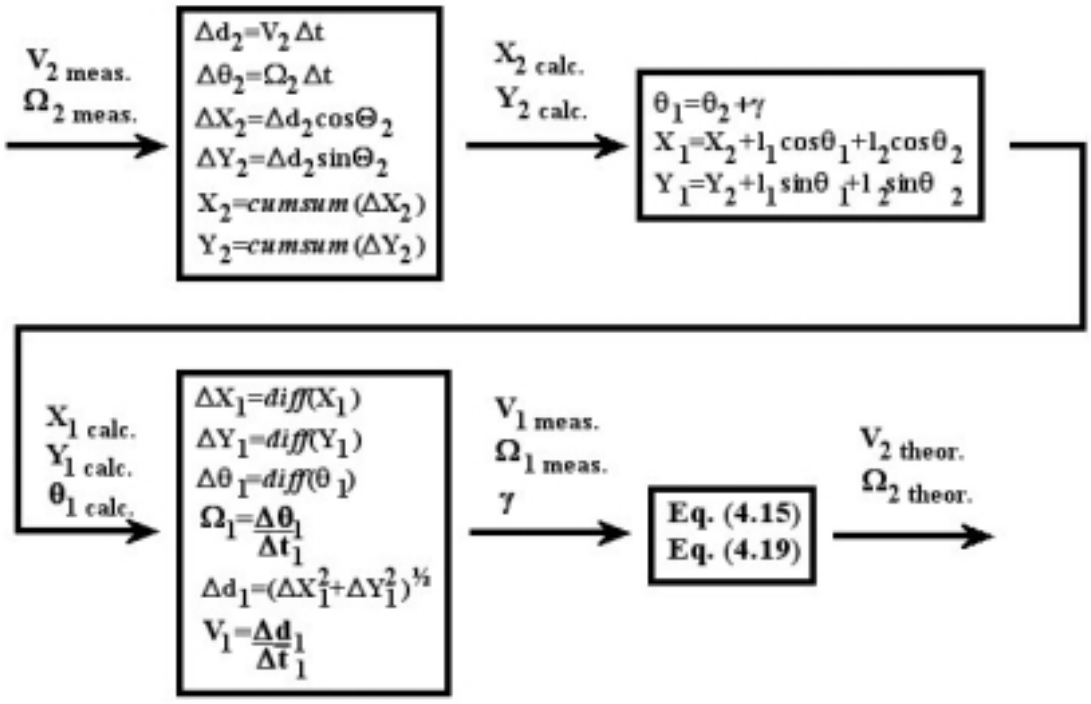

Figure 4. 8: Flow of calculations leading to generation of theoretically optimal values from experimental measurements.

4.2.1 Experimental Aspect

The technique used to reconstruct the experimental data was the 5th wheel speed and yaw technique described in **Section 3.3.3.3**. Using this technique, the trajectory of the trail vehicle center of mass was calculated. This set of data was used as part of the first set of plots. Also using the left cylinder displacement value, the trajectory of the lead vehicle center of mass was calculated. This was accomplished by using the trail vehicle's position and bearing as well as the angle, γ, to calculate the lead vehicle position and bearing by way of system geometry. This trajectory was used as part of the first set of plots, as well, but it also provided the speed and rotational velocity vectors of the lead vehicle center of mass. The speed and yaw vectors were calculated using the reverse of the techniques discussed in **Section 3.3.3**. The Cartesian coordinates were

broken down into time step-difference vectors. The Pythagorean theorem was then used to create the *Δdiff* vector, which was summed up. The derivative of this summation is the lead vehicle speed. The derivative of the bearing is the lead vehicle yaw. These two vectors, speed and yaw, are the key to correlation. As will be explained in the next section, speed and rotational velocity are the experimental quantities that are used to drive the theoretical calculations. The left cylinder displacement was also used to calculate the angle, γ, which was used in the second set of plots.

4.2.2 Theoretical Aspect

After the speed and rotational velocity vectors of the lead vehicle were calculated from the experimental data, the theoretical algorithms utilized them. The theoretical model had the same initial speed, rotational velocity, and γ angle as the system did for the experimental tests. Then, using the experimental calculated vectors and Eqs. (4.15) and (4.19), the optimal speed, rotational velocity, and γ angle vectors were calculated. The γ angle vector was used in the second set of plots to compare with the experimental γ angles. The new speed and rotational velocity vectors were used to calculate the path of the rear vehicle center of mass that was optimal in terms of stress reduction to the TVLA. This path was included in the first set of plots to compare the lead vehicle path and the trail vehicle path from the experiments.

4.2.3 Correlation Results

After the correlation program was completed, all eighteen test sets from 07/15/03 and 07/16/03 were processed. The results of the correlation program are presented on the

following pages. The figures are described and discussed where necessary. The path plots have circles and cross marks designating the start and finish, respectively, of the runs.

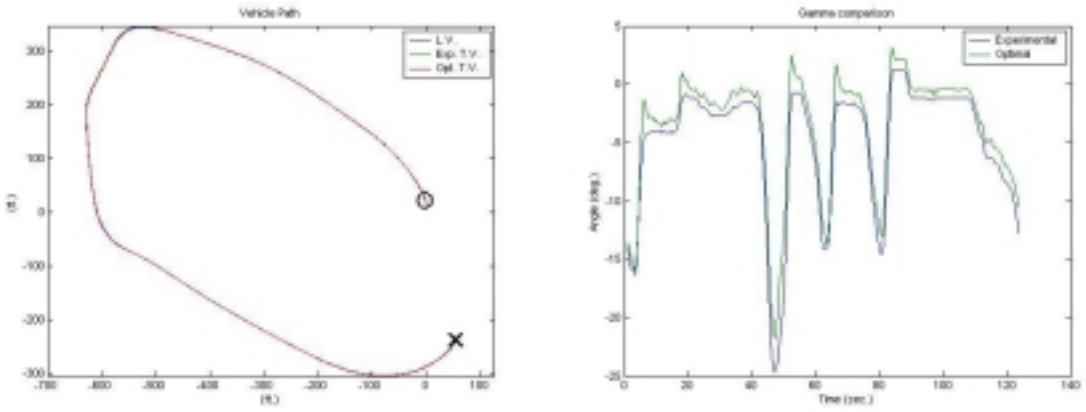

Figure 4. 9: The *corrcalc.m* results for *sabre2* from 07/15/03.

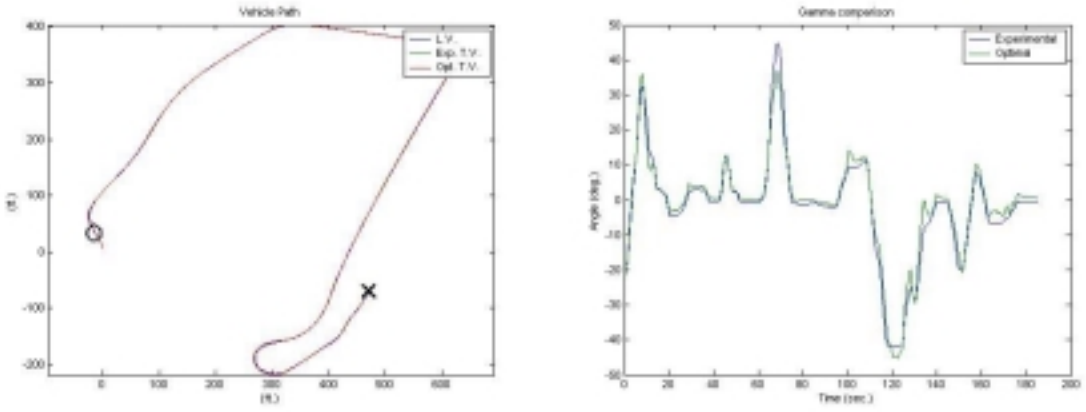

Figure 4. 10: The *corrcalc.m* results for *sabre3* from 07/15/03.

Figure 4.8 and **Figure 4.9** both cover tests that involved the driver moving around in an undirected path. This was referred to as meandering. In **Figure 4.8**, the vehicle system basically circled around to the left to end up behind its original position. The vehicles were constantly turning left. For this reason, the γ plot was always negative. As can be seen, the two curves differ. The optimal angle tends to be approximately one

to two degrees higher than the experimental data. On the steep declines and inclines of the downward spikes that are indicative of a sharp left turn, the optimal and experimental are very close to each other, only a 1-2 degree difference. In **Figure 4.9**, the optimal and experimental γ curves do not differ that much. The difference is less obvious than the last example. However, on the sharp upward spike, the experimental data reaches a higher angle and on the downward spike, optimal data reaches a greater magnitude.

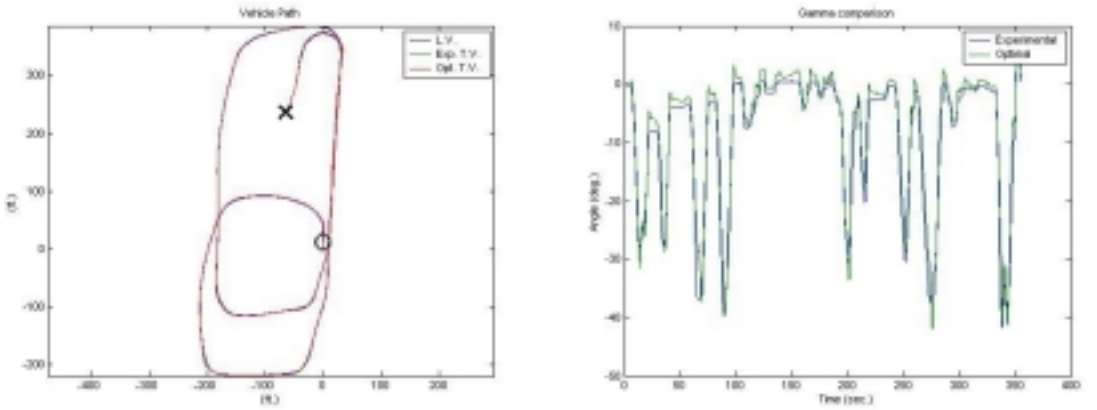

Figure 4. 11: The *corrcalc.m* results for *sabre4* from 07/15/03.

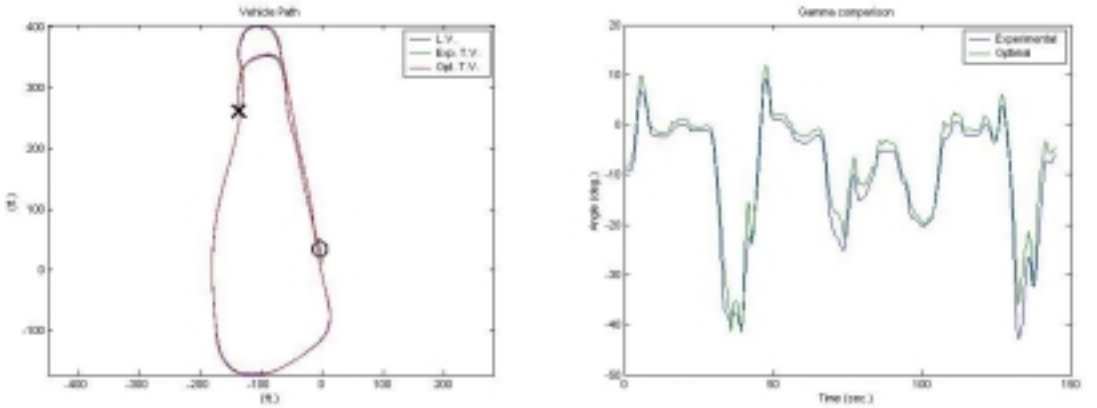

Figure 4. 12: The *corrcalc.m* results for *sabre5* from 07/15/03.

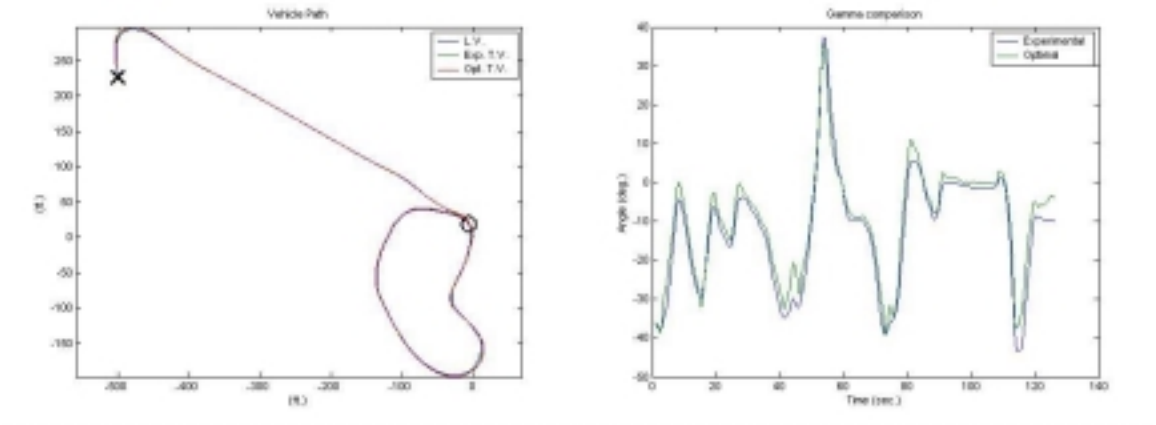

Figure 4. 13: The *corrcalc.m* results for *sabre6* from 07/15/03.

Figures 4.10, **4.11**, and **4.12** are all plots correlating narrowing cone test data. In the first two γ plots, the optimal value of γ tends to be slightly greater than the experimental values. The third plot does not have such a variance. However, the γ values tend to vary at the apex of spikes in the data. Sometimes, the experimental data has a greater magnitude. Sometimes, the experimental and optimal values are the same. Usually, though, the optimal data has a higher magnitude at these segments.

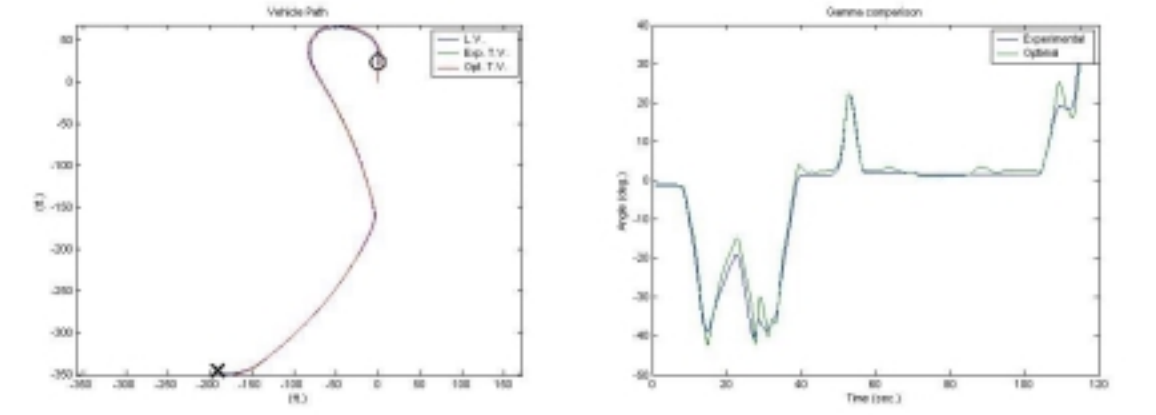

Figure 4. 14: The *corrcalc.m* results for *sabre7* from 07/15/03.

Figure 4.13 shows a maneuver that moved the vehicle system from the narrowing cone test to the slalom test. The experimental and optimal γ plots are very similar. Once again, the main differences occur during turns.

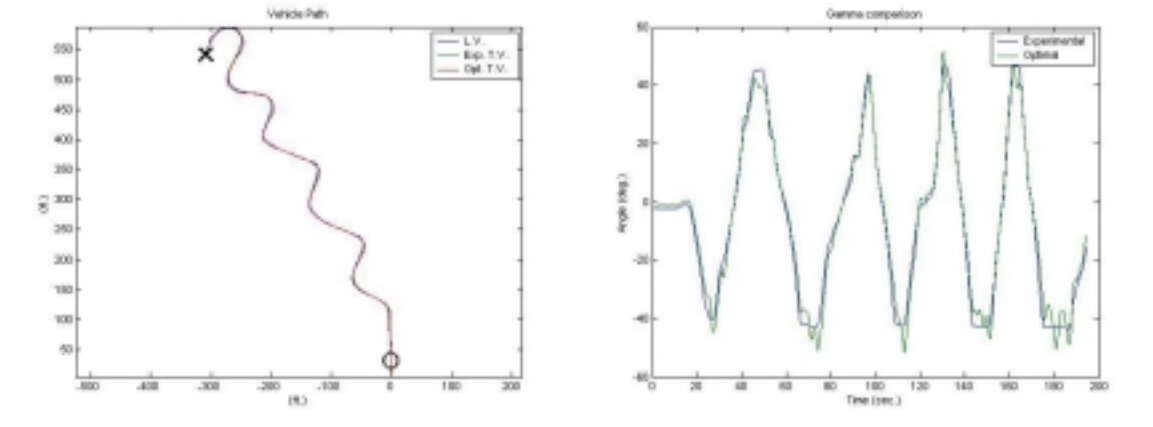

Figure 4. 15: The *corrcalc.m* results for *sabre8* from 07/15/03.

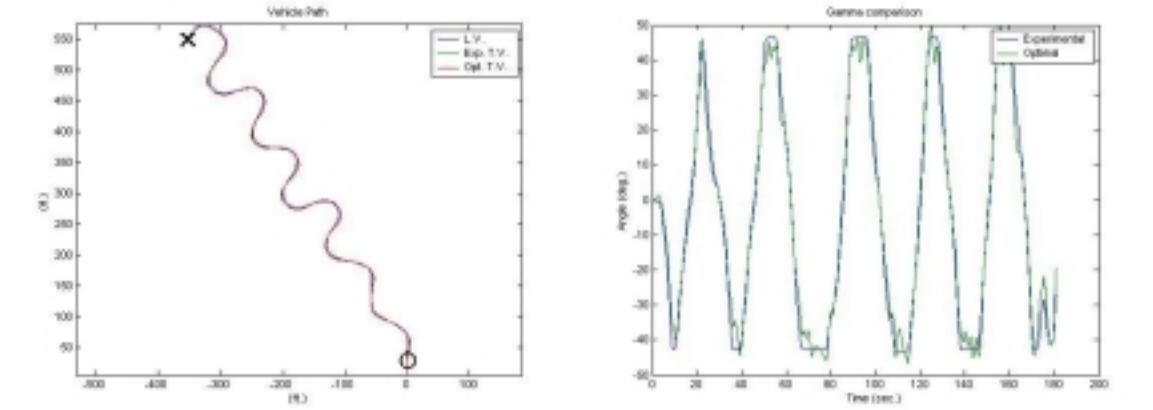

Figure 4. 16: The *corrcalc.m* results for *sabre9* from 07/15/03.

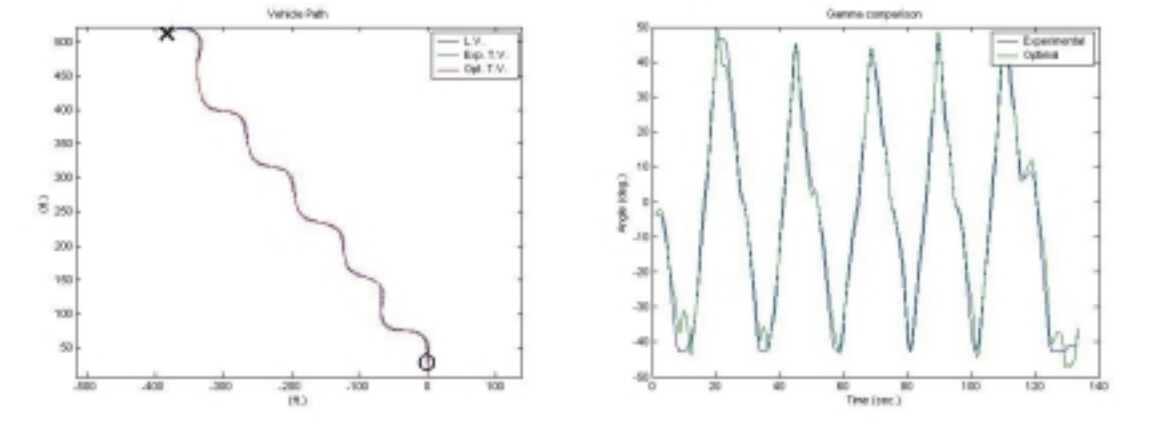

Figure 4. 17: The *corrcalc.m* results for *sabre10* from 07/15/03.

Figures 4.14, **4.15**, and **4.16** are slalom test plots. Each consecutive slalom run was an improvement over the last. This can be seen through the γ plots. The first has a sloppy cyclic pattern. The second is an improvement, and the last has fairly defined cycles. All three have well correlated γ values for the majority of time.

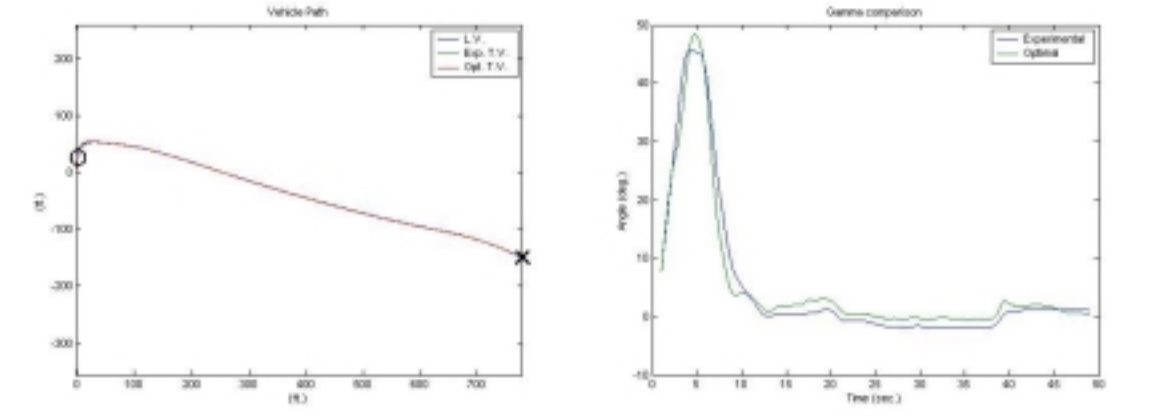

Figure 4. 18: The *corrcalc.m* results for *sabre11* from 07/15/03.

Figure 4.17 is a high-speed test plot. The experimental γ value is lower than the optimal value for the most of the run. However, for undetermined reasons, that difference diminishes during the last ten seconds of the run.

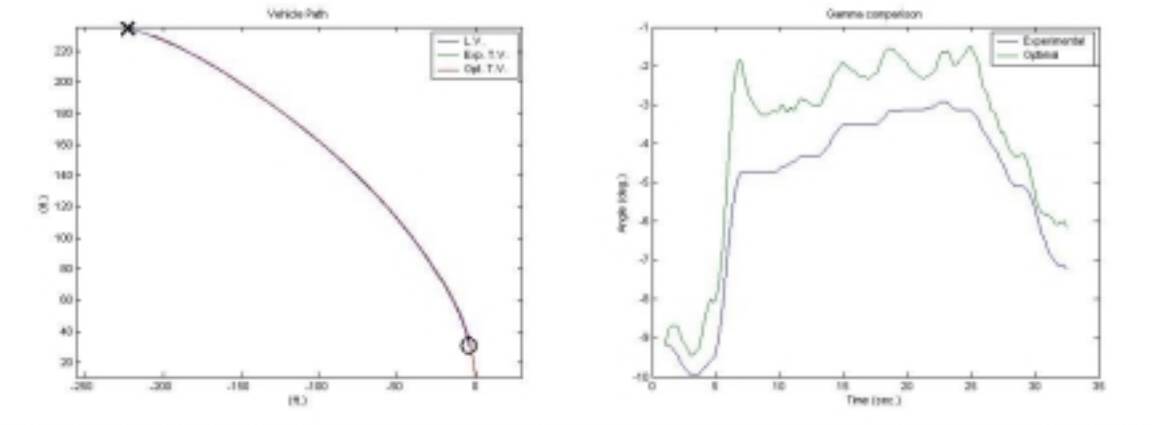

Figure 4. 19: The *corrcalc.m* results for *sabre1* from 07/16/03.

Another meandering run, this test was the first of the day. The γ plots appear to vary more than any as of yet seen. It should be noted that the range on the vertical axis is considerably smaller that previous figures. The difference between the two γ values plots averages between one and two degrees, which is comparable to previous plots.

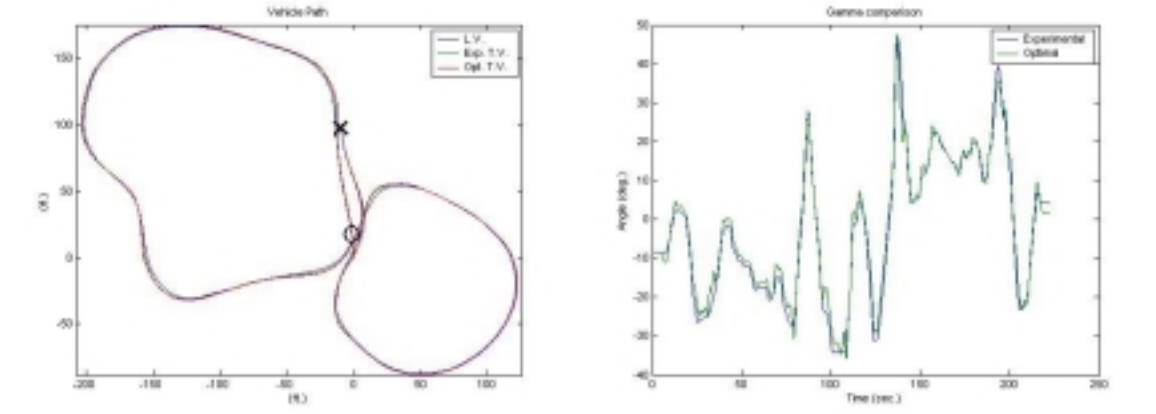

Figure 4. 20: The *corrcalc.m* results for *sabre2* from 07/16/03.

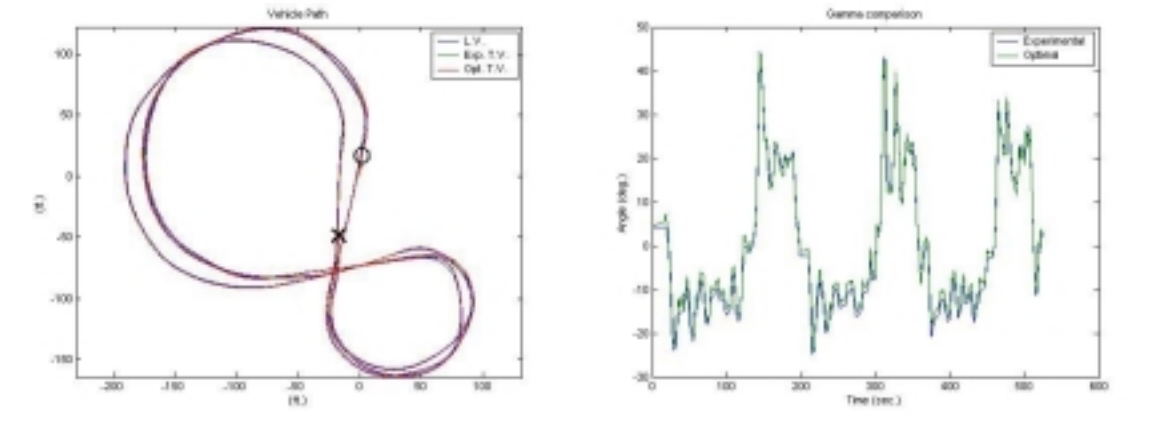

Figure 4. 21: The *corrcalc.m* results for *sabre3* from 07/16/03.

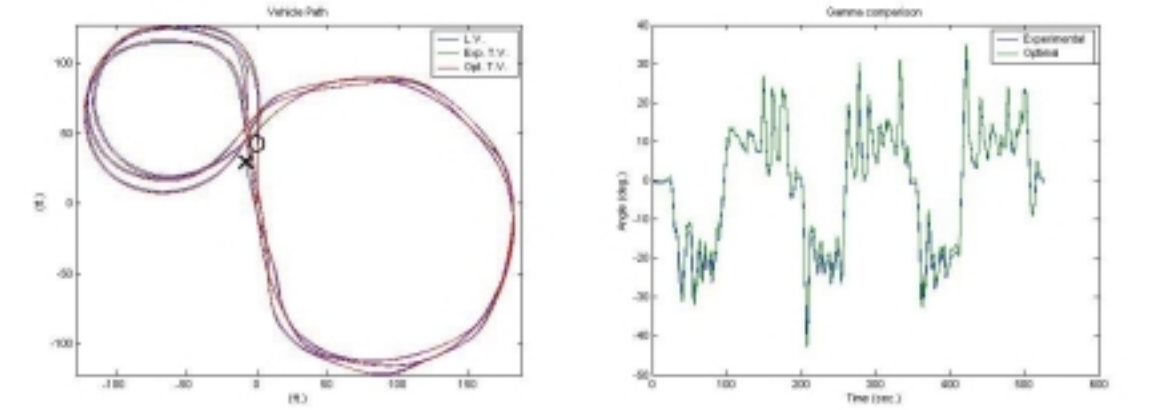

Figure 4. 22: The *corrcalc.m* results for *sabre4* from 07/16/03.

These three figures, **4.19**, **4.20**, and **4.21**, show the modified figure eight tests. The first run was very sloppy due to the nature of cone configuration #1. Nonetheless, the γ values are closely correlated for a majority of the run. There are some differences. Given the vertical axis scale, the differences appear to be close to those in **Figure 4.18**. The other plots show three cycles of the vehicles running the course. **Figure 4.20** has the same γ differences. These differences show up less, however, in **Figure 4.21**.

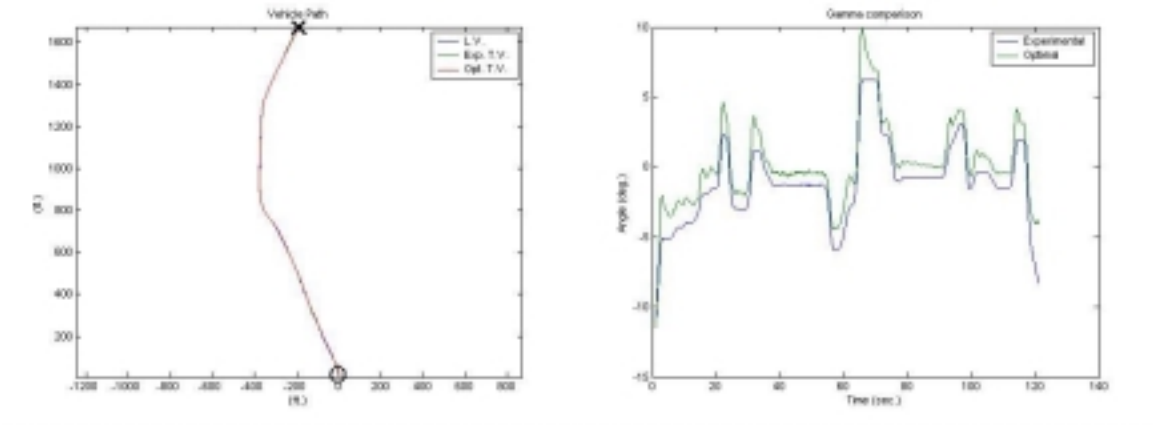

Figure 4. 23: The *corrcalc.m* results for *sabre5* from 07/16/03.

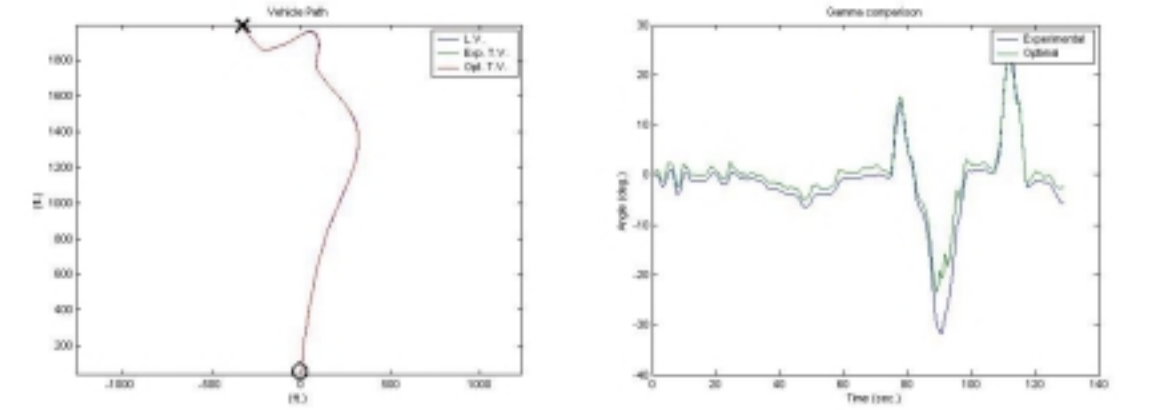

Figure 4. 24: The *corrcalc.m* results for *sabre6* from 07/16/03.

These two sets of plots, **4.22** and **4.23**, show the transit from the Spin-Up Packed Snow Area to the Ice Rink for the high-speed tests. A note about these runs is the fact that the vehicles were in manual control. The optimal γ value offset is present in both tests. There is very little visible difference in these paths compared to when the system is remote operated. The paths were, for the most part, fairly linear. At the end of the *sabre6* run in **Figure 4.23**, the vehicles make a slight right turn, followed by a sharp left turn, and end with a mild right turn. It is at the sharp left turn that the γ angle has the greatest discrepancy. The difference at this point is around ten degrees. Since this is the

only sharp turn made while in manual control, the inconsistency can most likely be attributed to non-optimal performance while controlling the vehicles.

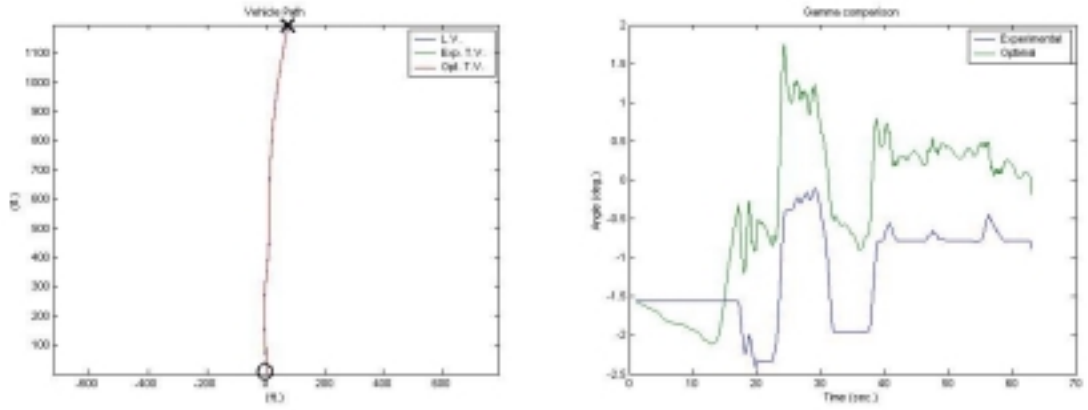

Figure 4. 25: The *corrcalc.m* results for *sabre7* from 07/16/03.

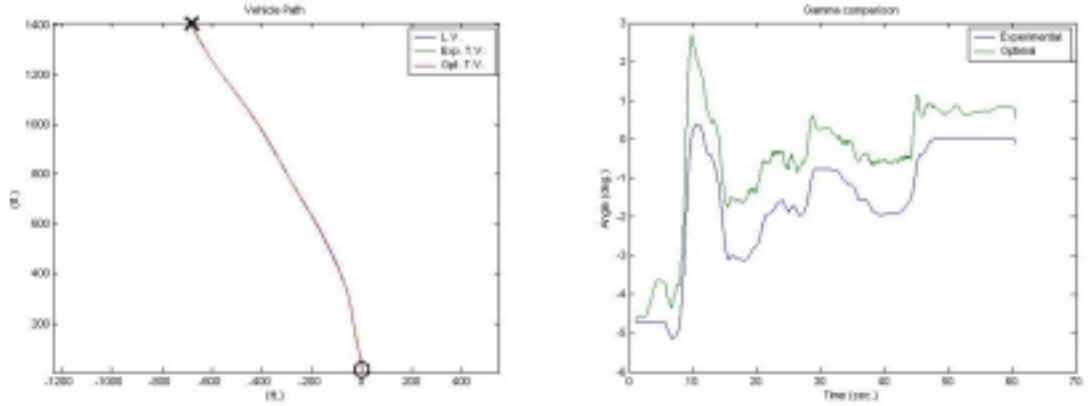

Figure 4. 26: The *corrcalc.m* results for *sabre8* from 07/16/03.

These, **4.24** and **4.25**, were the high-speed tests. The paths were very linear and because of this, there was very little variation in γ angles. That is why the offset looks inflated. The end of the data defined in **Figure 4.24** may be corrupted because it is near the time that the TVLA began to fail. Otherwise, this data has similar properties to the other runs.

Chapter 5. Conclusions

5.1 Conclusions

Overall, the theoretical calculated data correlates well with the experimental data. The biggest variations occurred at times when the system is midway through sharp turns. At these points, the operator made sharper turns than was calculated as optimal. The only other major deviation was a fairly consistent offset of about 1.5 degrees. The offset could be due to a higher pressure in one of the hydraulic lines than in the other. The TVLA hydraulic system is set up so the pressure line connected to the rod side of one of the side cylinders is on the same circuit as the line connected to the chamber side of the other side cylinder. If one of these pressure lines is different than the other, the resistant force in the cylinder may be just enough to keep the trail vehicle 1.5 degrees from its optimal position. This would also explain how the γ values match well during changes in rotation, because that change would have enough energy to overcome that resistance.

Less than a week of experimental testing was originally scheduled. Only three of these days were scheduled to contain tests that were viable for data acquisition. Concerns surfaced that there were not going to be enough tests to provide enough information for the project. These concerns were further compounded when the TVLA failed. The data were acquired and dissected to determine the test content of basic maneuvers. After this, it was concluded that the tests were adequate sources of these maneuvers.

Test Date	Left Turns	Right Turns	Straightaways	Left to Right S-curves	Right to Left S-curves
07/15/03	13	5	18	9	12
07/16/03	13	11	12	4	1
Total	26	16	30	13	13

Table 5. 1: Maneuver content of experimental data.

5.2 Recommendations

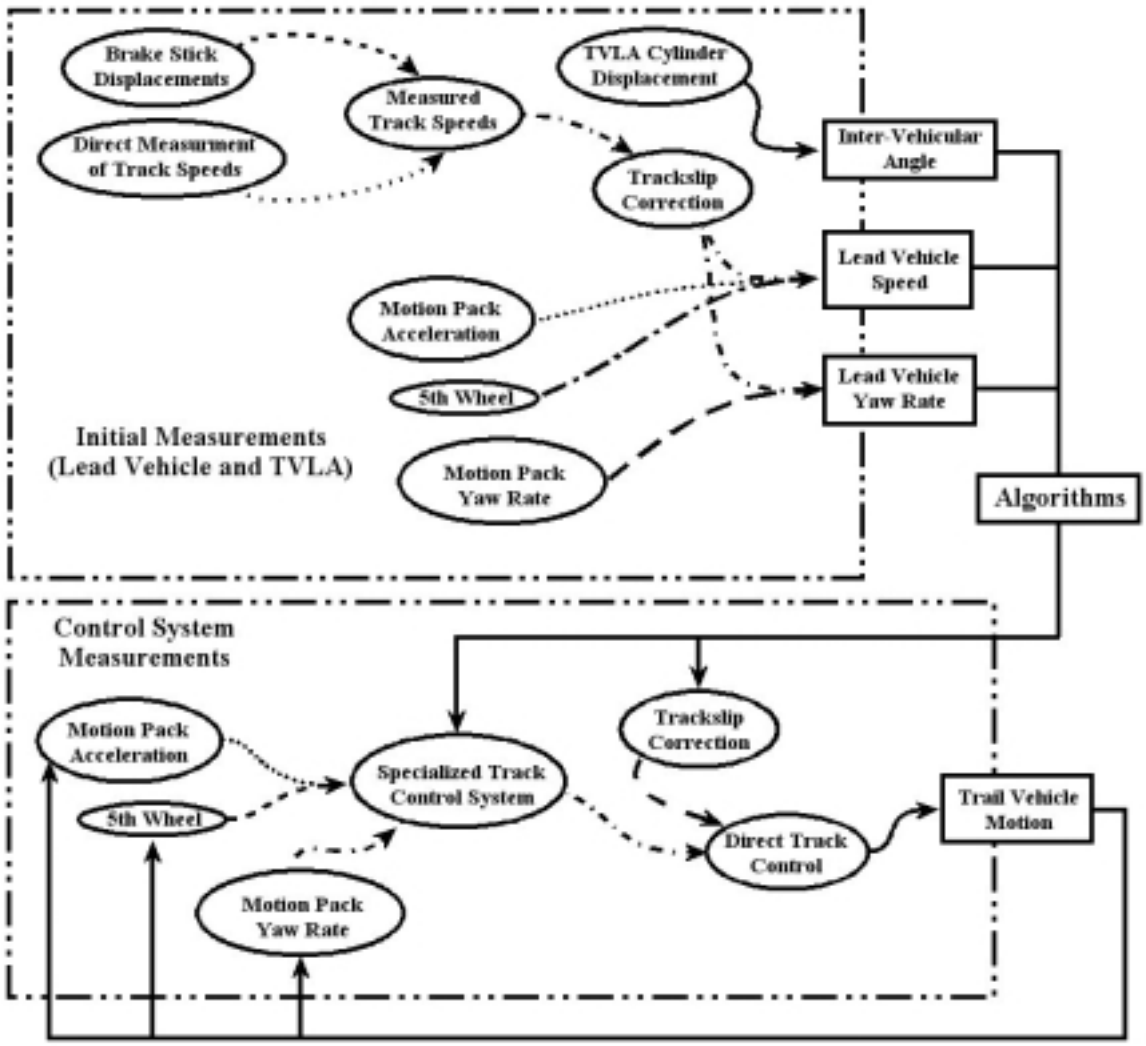

Figure 5. 1: Flow chart of several viable control techniques and related measurements.

This report marks the first step to automating the trail vehicle in the *Sabre/Panther Lite* setup. More steps will need to be taken to finish the project. The following describe some of the steps that will lead to project completion.

Control System – A control system will be needed to implement the algorithms developed in this report. There are several ways this control

system could work. Acquiring the lead vehicle motion can be made by monitoring the inputs to the system, via remote operation signals or brake stick displacements and throttle positions, or the speed and rotation can be measured directly. In a similar fashion, the control system could either calculate and implement necessary track speeds, or the trail vehicle motion could be measured and the track speeds would be controlled until the motion was as desired. The latter would remove the necessity to perform trackslip studies. The best choice will depend on which input measurement will produce the most accurate lead vehicle trajectory. **Figure 5.1** is a flow chart showing these possible methods of control.

Trackslip Studies – If track speed measurement or control is used, a trackslip study will have to be conducted. The study will have to develop a method for determining the *TS* coefficient for various ground type. The ground types will have to be categorized in great detail so there will be no need for ground sampling and testing on the battlefield before use.

Brake Stick Displacements and Throttle – If brake stick displacement and throttle are used to calculate the lead vehicle motion, added transducers would need to be installed to measure throttle position. For the tests described in this report, there were not enough resources to measure the throttle position. Without that, the brake stick displacements are less useful, as they calculate track speed as a percentage of throttle.

Additional Load – As a first step, the algorithms generated in this report calculate the optimal speed and rotation for the trail vehicle to follow the

lead vehicle. The main purpose for the *Sabre* development was to link the two M113s together because the work was too great for one of them to handle alone. Therefore, future work should contain more studies on how to modify the algorithms in order to add more force when necessary.

Three-Dimensional Analysis – One of the main assumptions used while measuring data and generating the algorithms was that the terrain could be considered two-dimensional. In future applications, it is feasible that the system will encounter terrain that is decidedly three-dimensional and the aforementioned assumption will be no longer valid. Therefore, more tests will need to be conducted to generate and validate 3D versions of these algorithms.

References

[1] "Transport Guidance: M113 Family of Vehicles," Technical Manual TM 55-2350-224-14, Washington D.C., February 1993

[2] Federation of American Scientists. "M113A1 Armored Personnel Carrier." *DOD 101*. 5 Feb 2000

[3] LTC David Ogg, “M113 FOV Overview”, Combat Vehicle Conference, Sep 1998

[4] Baladi, George Y., and Behzad Rohani, "Analysis of Steerability of Tracked Vehicles; Theoretical Predictions Versus Field Measurements," Final Report, U.S. Army Engineer Waterways Experiment Station - Structures Laboratory, Miscellaneous Paper SL-81-3, Vicksburg, Miss., March 1981

[5] Ahmedova, N.K., V.B. Kolmanovskii and A.I. Matasov, "Constructive Filtering Algorithms for Delayed Systems With Uncertain Statistics," *Journal of Dynamic Systems, measurement and Control.* June 2003, Vol. 125. pp. 229-235

[6] Winsauer, Sharon A., "A Program and Documentation for Simulation of a Tracked Feller/Buncher," U.S. Department of Agriculture - Forest Service, North Central Forest Experiment Station, St. Paul, Minn., 1980

[7] Ridley, Peter, and Peter Corke, "Load Haul Dump Vehicle Kinematics and Control," *Journal of Dynamic Systems, Measurement and Control,* March 2003

[8] Craig, John J. *Introduction to Robotics.* 2nd Ed. New York: Addison-Wesley, 1989

[9] Mabie, Hamilton H., and Charles F. Reinholtz. *Mechanisms and Dynamics of Machinery.* 2nd Ed. New York: Wiley, 1987

[10] Baruh, Haim. *Analytical Dynamics*. New York: McGraw-Hill, 1999

[11] A. Deshpande and J. Luntz." Enhancing mobility of a group of mobile robots via physical co-operation among the robots" In Proceedings of SPIE Conference on Un-manned Ground Vehicle Technology V, 2003.

Appendix A. Acronyms

Companies/Organizations

FMC	Food Machinery Corporation
JPO	Joint Programs Office
KRC	Keweenaw Research Center
MTU	Michigan Technological University
TACOM	Tank Automotive and Armaments Command
TARDEC	Tank-Automotive Research, Design, and Engineering Center
UDLP	United Defense Limited Partnership
VSIG	Vehicle Sensor Integration Group

Military

APC	Armored Personnel Carrier
CLAMS	Clearance Lane Marking System
CV	Command Vehicle
FOV	Family of Vehicles
HMMWV	High-Mobility Multi-purpose Wheeled Vehicle
TVLA	Tandem Vehicle Linkage Assembly

Electronic/Computer

ADC	Analog to Digital Converter
CSV	Comma Separated Variable
DAQ	Data Acquisition
MIPS	Million Instructions Per Second
MP	Measuring Ports
WSM	Wheel Speed Monitor

Other

CCW	Counter Clockwise
CW	Clockwise
SUPSA	Spin-up, Packed Snow Area

Appendix B. Instrumentation

Sensor Type / Brand	**Serial #**	**Location**
Wheel Speed Encoders		
Correvit	4 818 227 A	Lead Vehicle Left Sprocket
Correvit	4 806 368 A	Lead Vehicle Right Sprocket
Correvit	4 806 370 A	Trail Vehicle Left Sprocket
Correvit	4 806 365 A	Trail Vehicle Right Sprocket
2" String Pot		
UniMeasure	A26754	Lead Vehicle Left Differential Arm
UniMeasure	A18913	Lead Vehicle Right Differential Arm
10" String Pot		
UniMeasure	A25166	Trail Vehicle Left Differential Arm
UniMeasure	A27974	Trail Vehicle Right Differential Arm
24" String Pot		
UniMeasure	32090354	TVLA Vertical Cylinder
5' String Pot		
Research Inc.	2116	TVLA Left Horizontal Cylinder
Motion Pack		
Systron Donner	0198	Approx. Center of Mass of Trail Vehicle
Strain Type Pressure Sensor		
Sensotec	307753	MP1 of the TVLA Hydraulic System
Sensotec	307757	MP2 of the TVLA Hydraulic System
5th Wheel		
Labeco (Mod.)	1561	Off the Rear Right Side of the Trail Vehicle

Table B.1: Instrumentation List

Appendix C. Testing Information

Date (YYMMDD)	Name	Description
030711		
	Sabre1	Driving out to the Spin Up Packed Snow Area.
	Sabre2	Random driving around the SUPSA.
	Sabre3	Random driving around the SUPSA.
	Sabre4	Random driving around the SUPSA.
	Sabre5	Random driving around the SUPSA.
	Sabre6	Random driving around the SUPSA.
	Sabre7	Random driving around the SUPSA.
	Sabre8	Random driving around the SUPSA.
	Sabre9	Random driving around the SUPSA.
030715		
	Sabre2	Meandering. New camera position test.
	Sabre3	More meandering
	Sabre4	Narrowing Cone Test. 2x (Circles around first)
	Sabre5	Narrowing Cone Test. 2x
	Sabre6	Narrowing Cone Test. (Circles around first)
	Sabre7	Maneuvering to slalom cones.
	Sabre8	First slalom.
	Sabre9	Second slalom.
	Sabre10	Third slalom.
	Sabre11	High-Speed Test.
030716		
	Sabre1	Meandering.
	Sabre2	Modified Figure Eight Test. Large Loop CCW #1.
	Sabre3	Modified Figure Eight Test. Large Loop CCW #2,#3,#4. (#3 @ 200s) (#4 @ 365s)
	Sabre4	Modified Figure Eight Test. Large Loop CW #1,#2,#3. (#2 @ 195s) (#3 @ 355s)
	Sabre5	Maneuvering to ice track in manual mode.
	Sabre6	Maneuvering to ice track in manual mode.
	Sabre7	High-Speed Test #1. Up slope.
	Sabre8	High-Speed Test #2. Down slope.

Table C.1: Test Result File Names and Descriptions

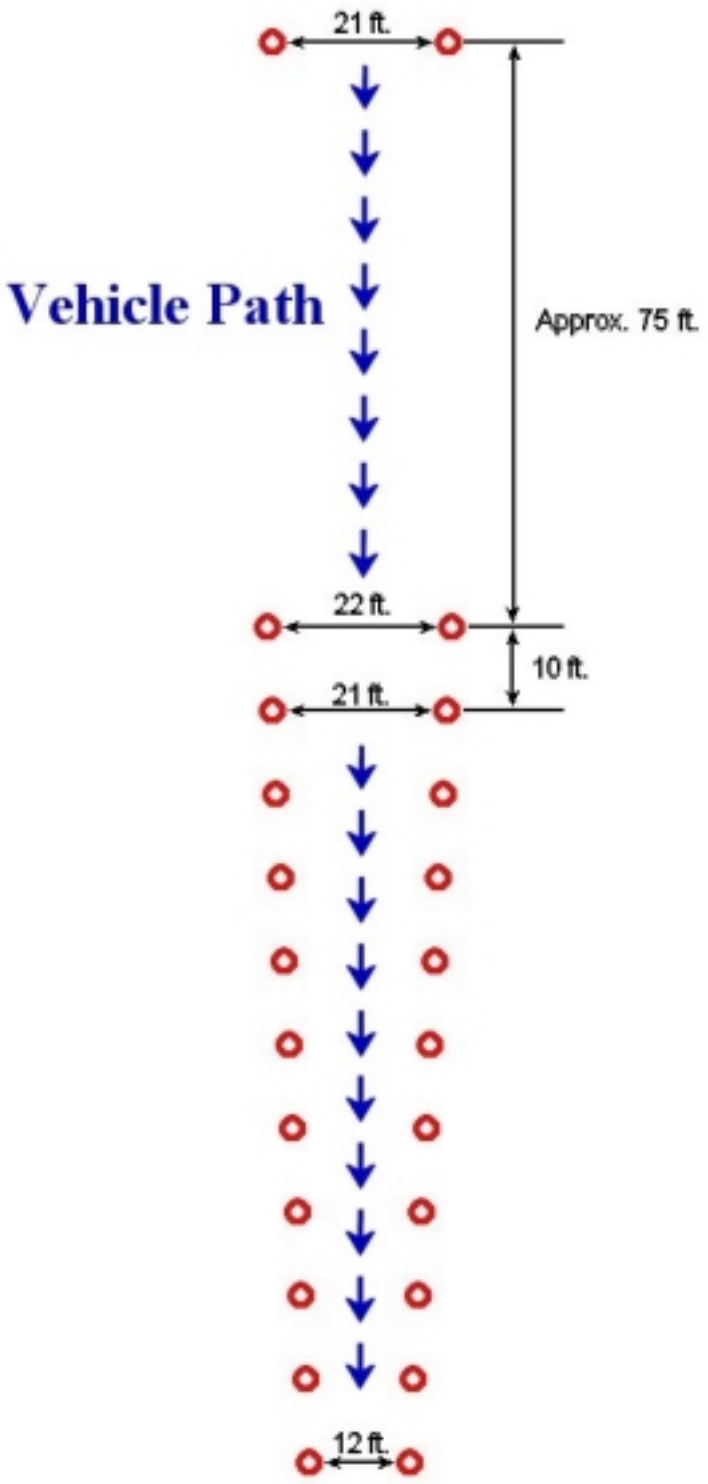

Figure C. 1: Cone layout for the narrowing cone test conducted on 07/15/03.

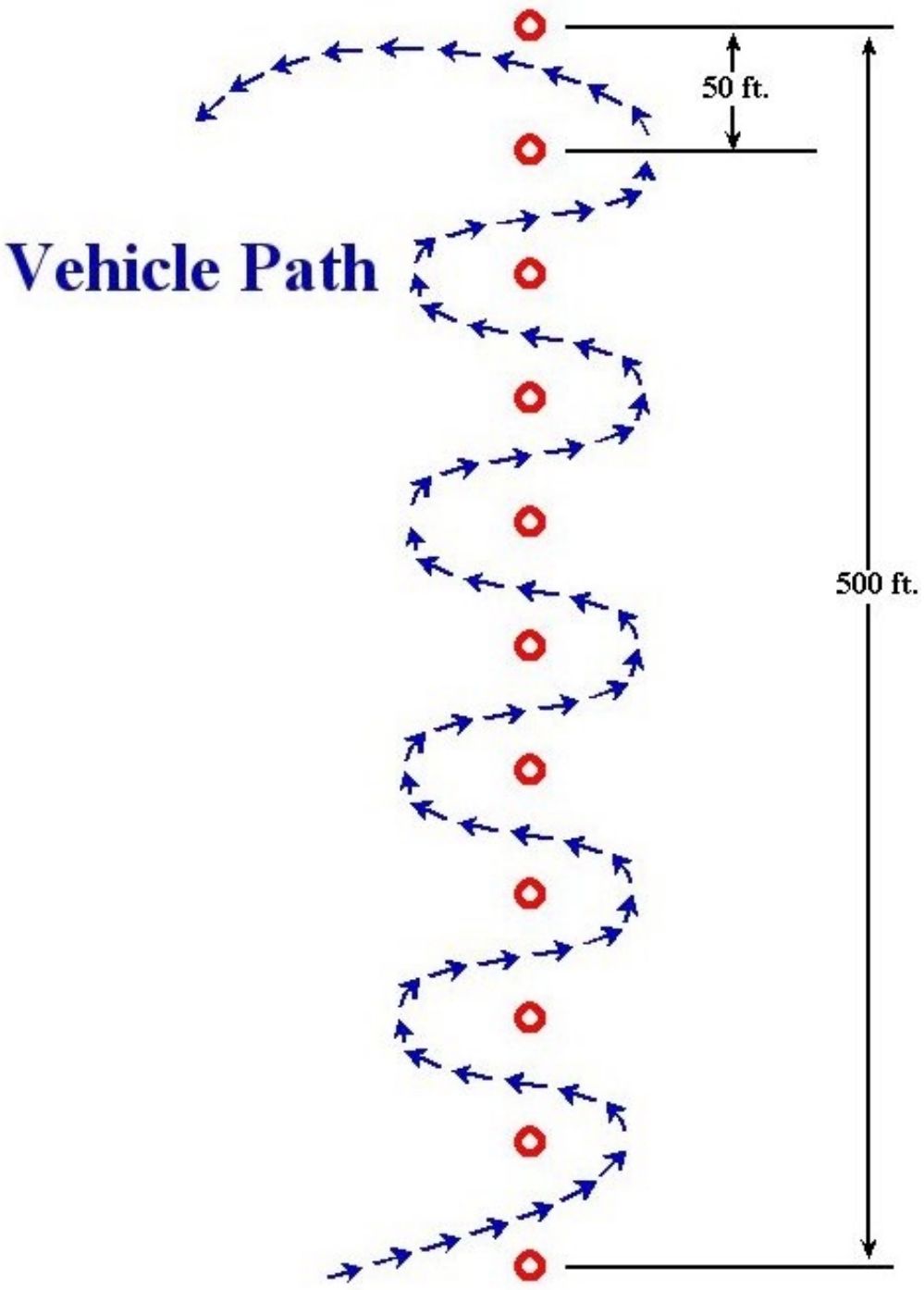

Figure C. 2: Cone layout of Slalom test conducted on 07/16/03.

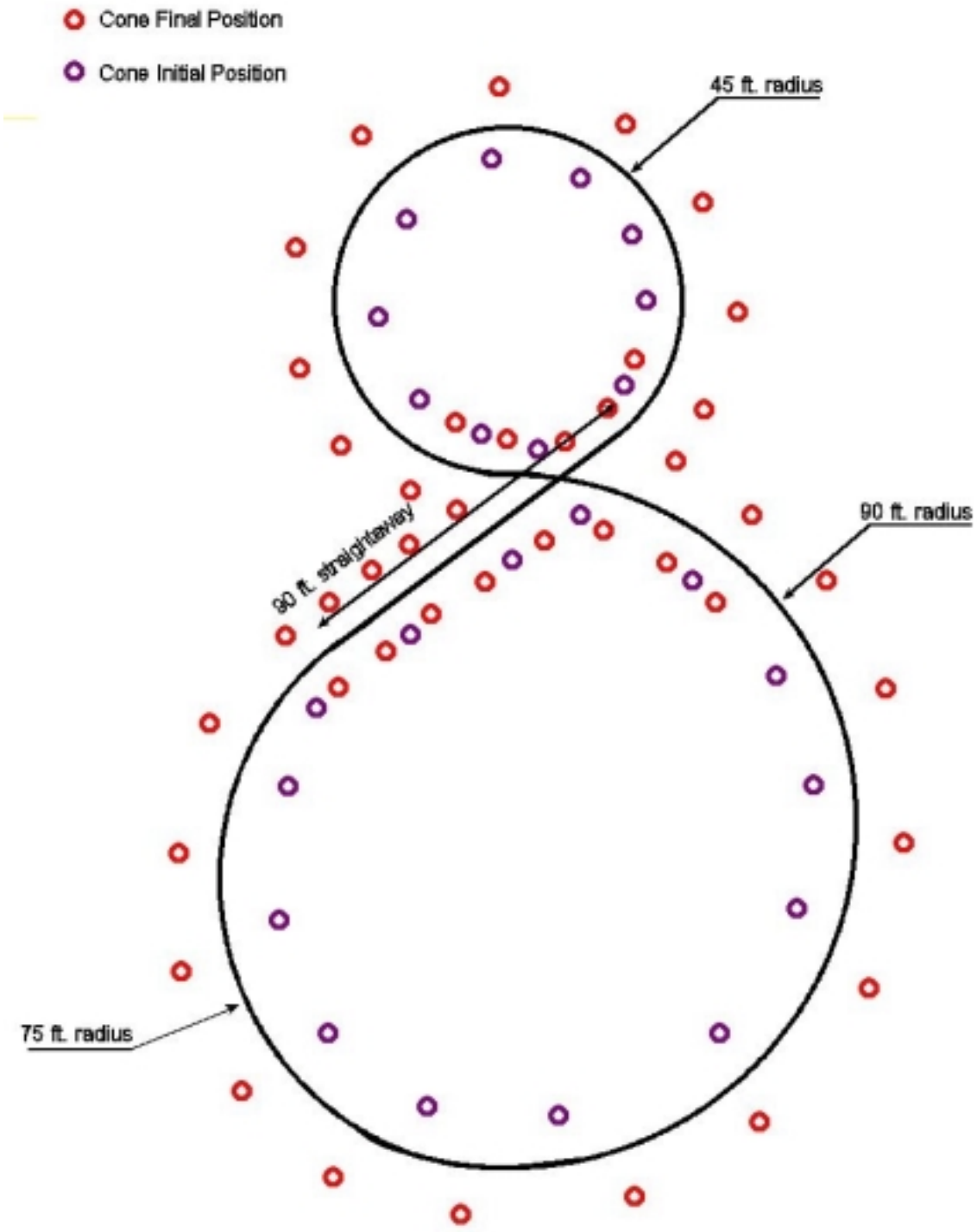

Figure C. 3: Cone layout for Modified Figure Eight test as conducted on 07/16/03.

Appendix D. Program Code

modcat.m

```
1   %MODCAT - File loader and separator.
2   %    Loads an ASCII file and separates the multi dimensional array
3   %  into individual vector components.  Corrects for the yaw rate
4   %  offset inherent in the test conducts between 07/11/03 and 07/16/03.
5   %  Also corrects speed error caused by track extension due to sprocket
6   %  rotational velocity.
7   %
8   %Ex.
9   %[dt,t,dist,speed,l1trk,r1trk,l2trk,r2trk,tdist,ldist,press1,press2,r1,l1,r2,l2,fwdbk,
    rtlft,updwn,roll,pitch,yaw,loaded_file,filt_size] = modcat('filename.asc',x);
10
11      function
    [dt,t,dist,speed,l1trk,r1trk,l2trk,r2trk,tdist,ldist,press1,press2,r1,l1,r2,l2,fwdbk,r
    tlft,updwn,roll,pitch,yaw,loaded_file,filt_size] = modcat(flnm,x)
12      if nargin==1
13          x=1;
14      end;
15
16      filt_size=x;
17      a=find(flnm=='.');
18      loaded_file=flnm(1:a-1);
19
20      m=dlmread(flnm);
21      t=m(:,1);
22      dist=m(:,2);
23      speed=m(:,3);
24      l1trk=1.0939.*m(:,4);   % Track over sprocket expansion factor
25      r1trk=1.0939.*m(:,5);   % Track over sprocket expansion factor
26      l2trk=1.0939.*m(:,6);   % Track over sprocket expansion factor
27      r2trk=1.0939.*m(:,7);   % Track over sprocket expansion factor
28      tdist=m(:,8);
29      ldist=m(:,9);
30      press1=m(:,10);
31      press2=m(:,11);
32      r1=m(:,12);
33      l1=m(:,13);
34      r2=m(:,14);
35      l2=m(:,15);
36      fwdbk=32.2*m(:,16);%+0.8936; %0.6882;
37      rtlft=32.2*m(:,17);%+0.6111;
38      updwn=32.2*m(:,18);%-.4985;
39      roll=m(:,19);
40      pitch=m(:,20);
41      yaw=m(:,21)-.24;  % Correction factor of the yaw zero.  Old 0.23
42      dt=mean(diff(t));
43
44      if x~=1
45         len=length(t);
46         a=filter(ones(1,x)/x,1,dist);
47         b=mean(dist(1:x)).*ones(size(dist(1:x)));
48         dist(1:x)=b;
49         dist(x:len)=a(x:len);
50         a=filter(ones(1,x)/x,1,speed);
51         b=mean(speed(1:x)).*ones(size(speed(1:x)));
52         speed(1:x)=b;
53         speed(x:len)=a(x:len);
54         a=filter(ones(1,x)/x,1,l1trk);
55         b=mean(l1trk(1:x)).*ones(size(l1trk(1:x)));
56         l1trk(1:x)=b;
57         l1trk(x:len)=a(x:len);
58         a=filter(ones(1,x)/x,1,r1trk);
59         b=mean(r1trk(1:x)).*ones(size(r1trk(1:x)));
60         r1trk(1:x)=b;
```

```
61        r1trk(x:len)=a(x:len);
62        a=filter(ones(1,x)/x,1,l2trk);
63        b=mean(l2trk(1:x)).*ones(size(l2trk(1:x)));
64        l2trk(1:x)=b;
65        l2trk(x:len)=a(x:len);
66        a=filter(ones(1,x)/x,1,r2trk);
67        b=mean(r2trk(1:x)).*ones(size(r2trk(1:x)));
68        r2trk(1:x)=b;
69        r2trk(x:len)=a(x:len);
70        a=filter(ones(1,x)/x,1,tdist);
71        b=mean(tdist(1:x)).*ones(size(tdist(1:x)));
72        tdist(1:x)=b;
73        tdist(x:len)=a(x:len);
74        a=filter(ones(1,x)/x,1,ldist);
75        b=mean(ldist(1:x)).*ones(size(ldist(1:x)));
76        ldist(1:x)=b;
77        ldist(x:len)=a(x:len);
78        a=filter(ones(1,x)/x,1,press1);
79        b=mean(press1(1:x)).*ones(size(press1(1:x)));
80        press1(1:x)=b;
81        press1(x:len)=a(x:len);
82        a=filter(ones(1,x)/x,1,press2);
83        b=mean(press2(1:x)).*ones(size(press2(1:x)));
84        press2(1:x)=b;
85        press2(x:len)=a(x:len);
86        a=filter(ones(1,x)/x,1,r1);
87        b=mean(r1(1:x)).*ones(size(r1(1:x)));
88        r1(1:x)=b;
89        r1(x:len)=a(x:len);
90        a=filter(ones(1,x)/x,1,l1);
91        b=mean(l1(1:x)).*ones(size(l1(1:x)));
92        l1(1:x)=b;
93        l1(x:len)=a(x:len);
94        a=filter(ones(1,x)/x,1,r2);
95        b=mean(r2(1:x)).*ones(size(r2(1:x)));
96        r2(1:x)=b;
97        r2(x:len)=a(x:len);
98        a=filter(ones(1,x)/x,1,l2);
99        b=mean(l2(1:x)).*ones(size(l2(1:x)));
100       l2(1:x)=b;
101       l2(x:len)=a(x:len);
102       a=filter(ones(1,x)/x,1,fwdbk);
103       b=mean(fwdbk(1:x)).*ones(size(fwdbk(1:x)));
104       fwdbk(1:x)=b;
105       fwdbk(x:len)=a(x:len);
106       a=filter(ones(1,x)/x,1,rtlft);
107       b=mean(rtlft(1:x)).*ones(size(rtlft(1:x)));
108       rtlft(1:x)=b;
109       rtlft(x:len)=a(x:len);
110       a=filter(ones(1,x)/x,1,updwn);
111       b=mean(updwn(1:x)).*ones(size(updwn(1:x)));
112       updwn(1:x)=b;
113       updwn(x:len)=a(x:len);
114       a=filter(ones(1,x)/x,1,roll);
115       b=mean(roll(1:x)).*ones(size(roll(1:x)));
116       roll(1:x)=b;
117       roll(x:len)=a(x:len);
118       a=filter(ones(1,x)/x,1,pitch);
119       b=mean(pitch(1:x)).*ones(size(pitch(1:x)));
120       pitch(1:x)=b;
121       pitch(x:len)=a(x:len);
122       a=filter(ones(1,x)/x,1,yaw);
123       b=mean(yaw(1:x)).*ones(size(yaw(1:x)));
124       yaw(1:x)=b;
125       yaw(x:len)=a(x:len);
126    end
```

allplot.m

```
1   %ALLPLOT - Plot Data.
2   %     Plots all of the data relative
3   %  to the first array defined.
4   %
5   %Ex.
6   %allplot(t,dist,speed,l1trk,r1trk,l2trk,r2trk,tdist,ldist,press1,press2,r1,l1,r2,l2,fw
    dbk,rtlft,updwn,roll,pitch,yaw);
7
8   function
    allplot(t,dist,speed,l1trk,r1trk,l2trk,r2trk,tdist,ldist,press1,press2,r1,l1,r2,l2,fwd
    bk,rtlft,updwn,roll,pitch,yaw)
9
10      close all;
11      figure('Name','5th Wheel',...
12          'NumberTitle','off',...
13          'Units','pixels',...
14          'Position',[0,0,1024,768],...
15          'Backingstore','off','Color','w');
16      movegui(gcf,'center');
17      subplot(2,1,1);
18      plot(t,dist);
19      xlabel('Time (s)');
20      ylabel('Distance (ft)');
21      title('5th Wheel Distance');
22      subplot(2,1,2);
23      plot(t,speed);
24      xlabel('Time (s)');
25      ylabel('Speed (mph)');
26      title('5th Wheel Speed');
27      figure('Name','Speeds',...
28          'NumberTitle','off',...
29          'Units','pixels',...
30          'Position',[0,0,1024,768],...
31          'Backingstore','off','Color','w');
32      movegui(gcf,'center');
33      plot(t,speed,t,l1trk,t,r1trk,t,l2trk,t,r2trk);
34      xlabel('Time (s)');
35      ylabel('Speed (mph)');
36      title('Speeds')
37      legend('5th Wheel','Lead Left','Lead Right','Trail Left','Trail Right',0);
38      figure('Name','Hydraulic Displacements',...
39          'NumberTitle','off',...
40          'Units','pixels',...
41          'Position',[0,0,1024,768],...
42          'Backingstore','off','Color','w');
43      movegui(gcf,'center');
44      plot(t,tdist,t,ldist);
45      xlabel('Time (s)');
46      ylabel('Displacement (in)');
47      title('Hydraulic Cylinder Displacements');
48      legend('Top Cylinder','Left Cylinder',0);
49      figure('Name','Hydraulic Pressures',...
50          'NumberTitle','off',...
51          'Units','pixels',...
52          'Position',[0,0,1024,768],...
53          'Backingstore','off','Color','w');
54      movegui(gcf,'center');
55      plot(t,press1,t,press2);
56      xlabel('Time (s)');
57      ylabel('Pressure (psi)');
58      title('Hydraulic Cylinder Pressures');
59      legend('MP1','MP2',0);
60      figure('Name','Brake Displacements',...
61          'NumberTitle','off',...
62          'Units','pixels',...
63          'Position',[0,0,1024,768],...
64          'Backingstore','off','Color','w');
```

```
    movegui(gcf,'center');
    plot(t,l1,t,r1,t,l2,t,r2);
    xlabel('Time (s)');
    ylabel('Displacement (in)');
    title('Braking String Pot Displacements');
    legend('Lead Left','Lead Right','Trail Left','Trail Right',0);
    figure('Name','Linear Accelerations',...
        'NumberTitle','off',...
        'Units','pixels',...
        'Position',[0,0,1024,768],...
        'Backingstore','off','Color','w');
    movegui(gcf,'center');
    plot(t,fwdbk,t,rtlft,t,updwn);
    xlabel('Time (s)');
    ylabel('Acceleration (g)');
    title('Motion Pack Linear Accelerations');
    legend('Fwd/Bk','Rt/Lft','Up/Dwn',0);
    figure('Name','Rotational Velocities',...
        'NumberTitle','off',...
        'Units','pixels',...
        'Position',[0,0,1024,768],...
        'Backingstore','off','Color','w');
    movegui(gcf,'center');
    plot(t,roll,t,pitch,t,yaw);
    xlabel('Time (s)');
    ylabel('Rotational Velocity (°/s)');
    title('Motion Pack Rotational Velocities');
    legend('Roll','Pitch','Yaw',0);
```

mplot4b.m

```
%MPLOT4B - M113 plotter.
%    Similar to MPLOT4, but compares the path generated
%  using the rear vehicle track speeds with the path generated
%  using the yaw rate and 5th Wheel speed.
%
% Ex.
% mplot4b(t,speed,yaw,l2trk,r2trk,10);


    function mplot4b(t,speed,yaw,l2trk,r2trk,skipnum,action);

    if nargin<5
        error('You must input the five variables!')
        return;
    elseif nargin<6
        skipnum=1;
        action='initialize';
    elseif nargin<7
        action='initialize';
    end;

    if strcmp(action,'initialize')


        figNumber=figure(...
            'Name','M113 Plotter (comparing SPEED and YAW with L2TRK and R2TRK)',...
            'NumberTitle','off',...
            'BackingStore','off',...
            'Units','normalized',...
            'Position',[0.1 0.1 0.8 0.8],...
            'Visible','off');
        colordef(figNumber,'black');
        axes('Units','normalized',...
            'Position',[0.05 0.15 0.75 0.80],...
            'Visible','off');
        textHndl=text(0,0,'Press the "Run" button to see the vehicle motion',...
```

```
37            'HorizontalAlignment','center');
38        axis([-1 1 -1 1]);
39
40        %==================================
41        % Information for button
42        labelColor=[0.8 0.8 0.8];
43        yInitPos=0.50;
44        xPos=0.85;
45        btnLen=0.10;
46        btnWid=0.10;
47        txtWid=0.05;
48        txtLen=0.2;
49        spacing=0.05;
50
51        %==================================
52        % The CONSOLE frame
53        frmBorder=0.02;
54        yPos=0.5-2*spacing-2.5*btnWid-frmBorder;
55        frmPos=[xPos-frmBorder yPos btnLen+2*frmBorder
  4*spacing+5*btnWid+2*(frmBorder)];
56        h=uicontrol('Style','frame',...
57            'Units','normalized',...
58            'Position',frmPos,...
59            'BackgroundColor',[0.5 0.5 0.5]);
60
61        %==================================
62        % The TIME box
63        btnNumber=1;
64        yPos=0.5+1.5*spacing+btnWid;
65        labelStr=['0.000 : ' num2str(max(t))];
66        cmdStr='run';
67
68        %Generic button information
69        btnPos=[.425-.1 0.05 txtLen txtWid];
70        timeHndl=uicontrol(...
71            'Style','edit',...
72            'Units','normalized',...
73            'Position',btnPos,...
74            'String',labelStr,...
75            'Enable','off');
76
77        %==================================
78        % The RUN button
79        btnNumber=2;
80        yPos=0.5+1.5*btnWid+2*spacing;
81        labelStr='Run';
82        cmdStr='run';
83        callbackstr='mplot4b(t,speed,yaw,l2trk,r2trk,1,''run'');';
84
85        %Generic button information
86        btnPos=[xPos yPos btnLen btnWid];
87        runHndl=uicontrol(...
88            'Style','pushbutton',...
89            'Units','normalized',...
90            'Position',btnPos,...
91            'String',labelStr,...
92            'Interruptible','on',...
93            'Callback',callbackstr);
94
95        %==================================
96        % The PAUSE button
97        btnNumber=3;
98        yPos=0.5+0.5*btnWid+spacing;
99        labelStr='Pause';
100       cmdStr='pause';
101       callbackstr='set(gca,''Userdata'',-1);';
102
103       %Generic button information
104       btnPos=[xPos yPos btnLen btnWid];
105       pauseHndl=uicontrol(...
106           'Style','pushbutton',...
```

```
107             'Units','normalized',...
108             'Position',btnPos,...
109             'String',labelStr,...
110             'Interruptible','on',...
111             'Callback',callbackstr,...
112             'Enable','off');
113 
114         %==================================
115         % The UNPAUSE button
116         btnNumber=4;
117         yPos=0.5-0.5*btnWid;
118         labelStr='Unpause';
119         cmdStr='unpause';
120         callbackstr='set(gca,''Userdata'',0);';
121 
122         %Generic button information
123         btnPos=[xPos yPos btnLen btnWid];
124         unpauseHndl=uicontrol(...
125             'Style','pushbutton',...
126             'Units','normalized',...
127             'Position',btnPos,...
128             'String',labelStr,...
129             'Interruptible','on',...
130             'Callback',callbackstr,...
131             'Enable','off');
132 
133         %==================================
134         % The STOP button
135         btnNumber=5;
136         yPos=0.5-1.5*btnWid-1*spacing;
137         labelStr='stop';
138         cmdStr='stop';
139         callbackstr='set(gca,''Userdata'',-3);';
140 
141         %Generic button information
142         btnPos=[xPos yPos btnLen btnWid];
143         stopHndl=uicontrol(...
144             'Style','pushbutton',...
145             'Units','normalized',...
146             'Position',btnPos,...
147             'String',labelStr,...
148             'Interruptible','on',...
149             'Callback',callbackstr,...
150             'Enable','off');
151 
152         %==================================
153         % The Close button
154         btnNumber=6;
155         yPos=0.5-2.5*btnWid-2*spacing;
156         labelStr='Close';
157         cmdStr='close';
158         callbackstr='close(gcf);';
159 
160         %Generic button information
161         btnPos=[xPos yPos btnLen btnWid];
162         closeHndl=uicontrol(...
163             'Style','pushbutton',...
164             'Units','normalized',...
165             'Position',btnPos,...
166             'String',labelStr,...
167             'Interruptible','on',...
168             'Callback',callbackstr);
169 
170         %==================================
171         % The Pivot Radius bar
172         btnNumber=7;
173         yPos=0.5+2.5*btnWid+3*spacing;
174         labelStr='50';
175         cmdStr='prad';
176         callbackstr='mplot4b(t,speed,yaw,l2trk,r2trk,1,''prad'');';
177 
```

```
178         %Generic button information
179         btnPos=[xPos yPos btnLen/2 txtWid/2];
180         pradHndl=uicontrol(...
181             'Style','edit',...
182             'Units','normalized',...
183             'Position',btnPos,...
184             'String',labelStr,...
185             'Interruptible','on',...
186             'Callback',callbackstr,...
187             'Userdata',50,...
188             'Enable','on');
189 
190         %Uncover the figure
191         hndlList=[timeHndl runHndl pauseHndl unpauseHndl stopHndl closeHndl pradHndl
    skipnum textHndl];
192         set(figNumber,'Visible','on',...
193             'UserData',hndlList);
194 
195         figure(figNumber);
196 
197 
198 
199 
200     elseif strcmp(action,'run')
201 
202         axHndl=gca;
203         figNumber=gcf;
204 
205         hndlList=get(figNumber,'UserData');
206         timeHndl=hndlList(1);
207         runHndl=hndlList(2);
208         pauseHndl=hndlList(3);
209         unpauseHndl=hndlList(4);
210         stopHndl=hndlList(5);
211         closeHndl=hndlList(6);
212         pradHndl=hndlList(7);
213         skipnum=hndlList(8);
214         textHndl=hndlList(9);
215 
216         set(pradHndl,'Enable','off');
217 
218         if strcmp('off',get(axHndl,'Visible'))
219             set(textHndl,'String','Processesing... Please wait.');
220         end;
221         prad=get(pradHndl,'Userdata');
222 
223         speed=speed.*88/60;
224         l2trk=l2trk.*88/60;
225         r2trk=r2trk.*88/60;
226 
227         speed2=(l2trk+r2trk)./2;
228 
229         % Calculates the angular position array.
230         L=size(t,1);
231         if L==1
232             L=size(t,2);
233         end;
234         yaw=pi.*yaw./180;
235 
236         theta=cumtrapz(t,yaw);
237         theta2=cumtrapz(t,(6./prad).*(l2trk-r2trk));
238 
239     %     for n=1:L;
240     %         while theta(n)>180
241     %             theta(n)=theta(n)-360;
242     %         end
243     %         while theta(n)<=-180
244     %             theta(n)=theta(n)+360;
245     %         end;
246     %     end;
247 
```

```
248
249
250         fpos=cumtrapz(t,speed);
251         fpos2=cumtrapz(t,speed2);
252
253
254         fpos=[fpos(1);diff(fpos)];
255         fpos2=[fpos2(1);diff(fpos2)];
256
257     %     for n=L:-1:2
258     %         fpos(n)=fpos(n)-fpos(n-1);
259     %         fpos2(n)=fpos2(n)-fpos2(n-1);
260     %     end;
261     %     cgx(1)=0;
262     %     cgy(1)=0;
263     %     cgx2(1)=0;
264     %     cgy2(1)=0;
265     %     for n=2:L;
266     %         set(progHndl,'String',[num2str(round(100*n/L)) '%']);
267     %         drawnow;
268     %
269     %         cgx(n)=cgx(n-1)+fpos(n)*sin(theta(n));
270     %         cgy(n)=cgy(n-1)+fpos(n)*cos(theta(n));
271     %         cgx2(n)=cgx2(n-1)+fpos2(n)*sin(theta2(n));
272     %         cgy2(n)=cgy2(n-1)+fpos2(n)*cos(theta2(n));
273     %
274     %     end;
275
276         cgx=cumsum(fpos.*sin(theta));
277         cgy=cumsum(fpos.*cos(theta));
278         cgx2=cumsum(fpos2.*sin(theta2));
279         cgy2=cumsum(fpos2.*cos(theta2));
280
281
282         xmin=min([min(cgx) min(cgx2)]);
283         xmax=max([max(cgx) max(cgx2)]);
284         ymin=min([min(cgy) min(cgy2)]);
285         ymax=max([max(cgy) max(cgy2)]);
286
287
288
289
290         set([runHndl closeHndl],'Enable','off');
291         set(timeHndl,'Enable','inactive');
292         set([pauseHndl stopHndl],'Enable','on');
293
294         set(axHndl,...
295             'XLim',[xmin xmax],'YLim',[ymin ymax],...
296             'Drawmode','fast',...
297             'Visible','on',...
298             'NextPlot','add');
299
300         cla;
301         head=line('color','r',...
302             'Marker','.',...
303             'markersize',25,...
304             'erase','xor',...
305             'xdata',cgx(1:50),'ydata',cgy(1:50));
306         body=line('color','y',...
307             'LineStyle','-',...
308             'erase','none',...
309             'xdata',[],'ydata',[]);
310         tail=line('color','b',...
311             'LineStyle','-',...
312             'erase','none',...
313             'xdata',[],'ydata',[]);
314         head2=line('color','w',...
315             'Marker','.',...
316             'markersize',25,...
317             'erase','xor',...
318             'xdata',cgx2(1:50),'ydata',cgy2(1:50));
```

```
319         body2=line('color','w',...
320             'LineStyle','-',...
321             'erase','none',...
322             'xdata',[],'ydata',[]);
323         tail2=line('color','w',...
324             'LineStyle','-',...
325             'erase','none',...
326             'xdata',[],'ydata',[]);
327         n=51;
328         while n<=L
329             x=get(gca,'Userdata');
330             set(head,'xdata',cgx(n),'ydata',cgy(n));
331             set(body,'xdata',cgx(n-50:n),'ydata',cgy(n-50:n));
332             set(tail,'xdata',cgx(1:n-50),'ydata',cgy(1:n-50));
333             set(head2,'xdata',cgx2(n),'ydata',cgy2(n));
334             set(body2,'xdata',cgx2(n-50:n),'ydata',cgy2(n-50:n));
335             set(tail2,'xdata',cgx2(1:n-50),'ydata',cgy2(1:n-50));
336             set(timeHndl,'String',[num2str(t(n)) ' : ' num2str(max(t))]);
337             drawnow;
338             while(x~=1)
339                 x=get(gca,'Userdata');
340                 switch x
341                     case -1
342                         set(pauseHndl,'Enable','off');
343                         set(unpauseHndl,'Enable','on');
344                         set(gca,'Userdata',-2);
345                     case 0
346                         set(unpauseHndl,'Enable','off');
347                         set(pauseHndl,'Enable','on');
348                         set(gca,'Userdata',1);
349                     case -3
350                         set(gca,'Userdata',1);
351                         n=L;
352                     otherwise
353                         pause(0.01);
354                 end;
355 
356             end;
357             n=n+skipnum;
358         end;
359         set([runHndl closeHndl pradHndl],'Enable','on');
360         set([timeHndl pauseHndl unpauseHndl stopHndl],'Enable','off');
361 
362 
363 
364     elseif strcmp(action,'prad')
365         axHndl=gca;
366         figNumber=gcf;
367 
368         hndlList=get(figNumber,'UserData');
369         timeHndl=hndlList(1);
370         runHndl=hndlList(2);
371         pauseHndl=hndlList(3);
372         unpauseHndl=hndlList(4);
373         stopHndl=hndlList(5);
374         closeHndl=hndlList(6);
375         pradHndl=hndlList(7);
376         skipnum=hndlList(8);
377         textHndl=hndlList(9);
378 
379         a=get(pradHndl,'Userdata');
380         b=get(pradHndl,'String');
381         c=str2double(b);
382         if (~isequalwithequalnans(NaN,c)&c>0)
383             set(pradHndl,'Userdata',c);
384         else
385             set(pradHndl,'String',num2str(a));
386         end;
387     end;
```

mplot5.m

```
1   %MPLOT5 - M113 plotter.
2   %    Similar to MPLOT4, but also adds
3   %  green outlines to represent the body
4   %  of the two M113's and a yellow triangle
5   %  to represent the TVLA.
6   %
7   % Ex.
8   % MPLOT5(t,speed,ldist,yaw,frmskp);
9
10
11      function mplot5(t,speed,ldist,yaw,skipnum,action);
12
13      if nargin<4
14          error('You must input the four variables!')
15          return;
16      elseif nargin<5
17          skipnum=1;
18          action='initialize';
19      elseif nargin<6
20          action='initialize';
21      end;
22
23      if strcmp(action,'initialize')
24
25
26          figNumber=figure(...
27              'Name','M113 Plotter (using SPEED for speed and YAW for yaw rate)',...
28              'NumberTitle','off',...
29              'BackingStore','off',...
30              'Units','normalized',...
31              'Position',[0.1 0.1 0.8 0.8],...
32              'Visible','off');
33          colordef(figNumber,'black');
34          axes('Units','normalized',...
35              'Position',[0.05 0.15 0.75 0.80],...
36              'Visible','off');
37          textHndl=text(0,0,'Press the "Run" button to see the vehicle motion',...
38              'HorizontalAlignment','center');
39          axis([-1 1 -1 1]);
40
41          %===================================
42          % Information for button
43          labelColor=[0.8 0.8 0.8];
44          yInitPos=0.50;
45          xPos=0.85;
46          btnLen=0.10;
47          btnWid=0.10;
48          txtWid=0.05;
49          txtLen=0.2;
50          spacing=0.05;
51
52          %===================================
53          % The CONSOLE frame
54          frmBorder=0.02;
55          yPos=0.5-2*spacing-2.5*btnWid-frmBorder;
56          frmPos=[xPos-frmBorder yPos btnLen+2*frmBorder
    4*spacing+5*btnWid+2*(frmBorder)];
57          h=uicontrol('Style','frame',...
58              'Units','normalized',...
59              'Position',frmPos,...
60              'BackgroundColor',[0.5 0.5 0.5]);
61
62          %===================================
63          % The TIME box
64          btnNumber=1;
65          yPos=0.5+1.5*spacing+btnWid;
```

```
66          labelStr=['0.000 : ' num2str(max(t))];
67          cmdStr='run';
68
69          %Generic button information
70          btnPos=[.425-.1 0.05 txtLen txtWid];
71          timeHndl=uicontrol(...
72              'Style','edit',...
73              'Units','normalized',...
74              'Position',btnPos,...
75              'String',labelStr,...
76              'Enable','off');
77
78          %===================================
79          % The RUN button
80          btnNumber=2;
81          yPos=0.5+1.5*btnWid+2*spacing;
82          labelStr='Run';
83          cmdStr='run';
84          callbackstr='mplot5(t,speed,ldist,yaw,1,''run'');';
85
86          %Generic button information
87          btnPos=[xPos yPos btnLen btnWid];
88          runHndl=uicontrol(...
89              'Style','pushbutton',...
90              'Units','normalized',...
91              'Position',btnPos,...
92              'String',labelStr,...
93              'Interruptible','on',...
94              'Callback',callbackstr);
95
96          %===================================
97          % The PAUSE button
98          btnNumber=3;
99          yPos=0.5+0.5*btnWid+spacing;
100         labelStr='Pause';
101         cmdStr='pause';
102         callbackstr='set(gca,''Userdata'',-1);';
103
104         %Generic button information
105         btnPos=[xPos yPos btnLen btnWid];
106         pauseHndl=uicontrol(...
107             'Style','pushbutton',...
108             'Units','normalized',...
109             'Position',btnPos,...
110             'String',labelStr,...
111             'Interruptible','on',...
112             'Callback',callbackstr,...
113             'Enable','off');
114
115         %===================================
116         % The UNPAUSE button
117         btnNumber=4;
118         yPos=0.5-0.5*btnWid;
119         labelStr='Unpause';
120         cmdStr='unpause';
121         callbackstr='set(gca,''Userdata'',0);';
122
123         %Generic button information
124         btnPos=[xPos yPos btnLen btnWid];
125         unpauseHndl=uicontrol(...
126             'Style','pushbutton',...
127             'Units','normalized',...
128             'Position',btnPos,...
129             'String',labelStr,...
130             'Interruptible','on',...
131             'Callback',callbackstr,...
132             'Enable','off');
133
134         %===================================
135         % The STOP button
136         btnNumber=5;
```

```
137         yPos=0.5-1.5*btnWid-1*spacing;
138         labelStr='stop';
139         cmdStr='stop';
140         callbackstr='set(gca,''Userdata'',-3);';
141 
142         %Generic button information
143         btnPos=[xPos yPos btnLen btnWid];
144         stopHndl=uicontrol(...
145             'Style','pushbutton',...
146             'Units','normalized',...
147             'Position',btnPos,...
148             'String',labelStr,...
149             'Interruptible','on',...
150             'Callback',callbackstr,...
151             'Enable','off');
152 
153         %==================================
154         % The Close button
155         btnNumber=6;
156         yPos=0.5-2.5*btnWid-2*spacing;
157         labelStr='Close';
158         cmdStr='close';
159         callbackstr='close(gcf);';
160 
161         %Generic button information
162         btnPos=[xPos yPos btnLen btnWid];
163         closeHndl=uicontrol(...
164             'Style','pushbutton',...
165             'Units','normalized',...
166             'Position',btnPos,...
167             'String',labelStr,...
168             'Interruptible','on',...
169             'Callback',callbackstr);
170 
171         %==================================
172         % The PROGRESS bar
173         btnNumber=7;
174         yPos=0.5+2.5*btnWid+3*spacing;
175         labelStr='0%';
176         cmdStr='progress';
177 
178         %Generic button information
179         btnPos=[xPos yPos btnLen txtWid];
180         progHndl=uicontrol(...
181             'Style','edit',...
182             'Units','normalized',...
183             'Position',btnPos,...
184             'String',labelStr,...
185             'Interruptible','on',...
186             'Enable','off');
187 
188         %Uncover the figure
189         hndlList=[timeHndl runHndl pauseHndl unpauseHndl stopHndl closeHndl progHndl
    skipnum textHndl];
190         set(figNumber,'Visible','on',...
191             'UserData',hndlList);
192 
193         figure(figNumber);
194 
195 
196 
197 
198     elseif strcmp(action,'run')
199 
200         axHndl=gca;
201         figNumber=gcf;
202 
203         hndlList=get(figNumber,'UserData');
204         timeHndl=hndlList(1);
205         runHndl=hndlList(2);
206         pauseHndl=hndlList(3);
```

```
207         unpauseHndl=hndlList(4);
208         stopHndl=hndlList(5);
209         closeHndl=hndlList(6);
210         progHndl=hndlList(7);
211         skipnum=hndlList(8);
212         textHndl=hndlList(9);
213
214         if strcmp('off',get(axHndl,'Visible'))
215             set(textHndl,'String','Processing...  Please Wait.');
216         end
217
218         x1=73/12;  %Distance from front of vehicle to CM.
219         x2=101/12; %Distance from rear of vehicle to CM.
220         x3=50/12;  %Distance from left of vehicle to CM.
221         x4=50/12;  %Distance from right of vehicle to CM.
222
223         speed=speed.*88/60;
224
225         % Calculates the angular position array.
226         L=size(t,1);
227         if L==1
228             L=size(t,2);
229         end;
230         theta=cumtrapz(t,yaw);
231         theta=theta.*pi./180;
232
233         %TVLA Constants
234         L1=sqrt(4.10^2+2.05^2);
235         alpha_o=atan(2);
236         gamma=(pi/2)-alpha_o;
237
238
239         fpos=cumtrapz(t,speed);
240         for n=L:-1:2
241             fpos(n)=fpos(n)-fpos(n-1);
242         end;
243         cgx(1)=0;
244         cgy(1)=0;
245
246         rbodyx(1,:)=[-x4 -x4 x3 x3 -x4];
247         rbodyy(1,:)=[x1 -x2 -x2 x1 x1];
248
249         set(progHndl,'Enable','inactive');
250     %     for n=2:L;
251     %         set(progHndl,'String',[num2str(round(100*n/L)) '%']);
252     %         drawnow;
253     % %         fpos(n)=.5*(fvel(n-1)+fvel(n)).*(t(n)-t(n-1));
254     %
255     %         cgx(n)=cgx(n-1)+fpos(n)*sin(theta(n));
256     %         cgy(n)=cgy(n-1)+fpos(n)*cos(theta(n));
257     %
258     %         c1x=cgx(n)-x3*cos(theta(n))+x1*sin(theta(n));
259     %         c1y=cgy(n)+x3*sin(theta(n))+x1*cos(theta(n));
260     %         c2x=cgx(n)+x4*cos(theta(n))+x1*sin(theta(n));
261     %         c2y=cgy(n)-x4*sin(theta(n))+x1*cos(theta(n));
262     %         c3x=cgx(n)+x4*cos(theta(n))-x2*sin(theta(n));
263     %         c3y=cgy(n)-x4*sin(theta(n))-x2*cos(theta(n));
264     %         c4x=cgx(n)-x3*cos(theta(n))-x2*sin(theta(n));
265     %         c4y=cgy(n)+x3*sin(theta(n))-x2*cos(theta(n));
266     %
267     %         rbodyx(n,:)=[c1x c2x c3x c4x c1x];
268     %         rbodyy(n,:)=[c1y c2y c3y c4y c1y];
269     %     end;
270
271         cgx=cumsum(fpos.*sin(theta));
272         cgy=cumsum(fpos.*cos(theta));
273
274
275         set(progHndl,'Enable','off');
276         xmin=min(cgx);
277         xmax=max(cgx);
```

```
278         ymin=min(cgy);
279         ymax=max(cgy);
280         x_sep=xmax-xmin;
281         y_sep=ymax-ymin;
282         sep=max([x_sep 1200*y_sep/907]);
283         x_diff=(sep-x_sep)/2;
284         y_diff=((907*sep/1200)-y_sep)/2;
285         xmin=xmin-x_diff;
286         xmax=xmax+x_diff;
287         ymin=ymin-y_diff;
288         ymax=ymax+y_diff;
289
290         set([runHndl closeHndl],'Enable','off');
291         set(timeHndl,'Enable','inactive');
292         set([pauseHndl stopHndl],'Enable','on');
293
294         set(axHndl,...
295             'XLim',[xmin xmax],'YLim',[ymin ymax],...
296             'Drawmode','fast',...
297             'Visible','on',...
298             'NextPlot','add');
299
300         cla;
301         head=line('color','r',...
302             'Marker','.',...
303             'markersize',25,...
304             'erase','xor',...
305             'xdata',cgx(1:50),'ydata',cgy(1:50));
306         body=line('color','y',...
307             'LineStyle','-',...
308             'erase','none',...
309             'xdata',[],'ydata',[]);
310         tail=line('color','b',...
311             'LineStyle','-',...
312             'erase','none',...
313             'xdata',[],'ydata',[]);
314         rm113=line('color','g',...
315             'LineStyle','-',...
316             'erase','xor',...
317             'xdata',[],'ydata',[]);
318         tvla=line('color','y',...
319             'LineStyle','-',...
320             'erase','xor',...
321             'xdata',[],'ydata',[]);
322         fm113=line('color','g',...
323             'LineStyle','-',...
324             'Erase','xor',...
325             'xdata',[],'ydata',[]);
326
327         n=51;
328         while n<=L
329             x=get(gca,'Userdata');
330             set(head,'xdata',cgx(n),'ydata',cgy(n));
331             set(body,'xdata',cgx(n-50:n),'ydata',cgy(n-50:n));
332             set(tail,'xdata',cgx(1:n-50),'ydata',cgy(1:n-50));
333             set(timeHndl,'String',[num2str(t(n)) ' : ' num2str(max(t))]);
334             set(rm113,...
335                 'xdata',[cgx(n)-x4*cos(theta(n))+x1*sin(theta(n)),...
336                     cgx(n)+x4*cos(theta(n))+x1*sin(theta(n)),...
337                     cgx(n)+x4*cos(theta(n))-x1*sin(theta(n)),...
338                     cgx(n)-x4*cos(theta(n))-x1*sin(theta(n)),...
339                     cgx(n)-x4*cos(theta(n))+x1*sin(theta(n))],...
340                 'ydata',[cgy(n)+x4*sin(theta(n))+x1*cos(theta(n)),...
341                     cgy(n)-x4*sin(theta(n))+x1*cos(theta(n)),...
342                     cgy(n)-x4*sin(theta(n))-x1*cos(theta(n)),...
343                     cgy(n)+x4*sin(theta(n))-x1*cos(theta(n)),...
344                     cgy(n)+x4*sin(theta(n))+x1*cos(theta(n))]);
345             theta2=acos((L1^2+1.25^2-(sqrt(4.1^2+0.80^2)+(ldist(n)-
    32.7)/12)^2)/(2.5*L1))-alpha_o;
346             TVLA1x=cgx(n)+6.09*sin(theta(n));
347             TVLA1y=cgy(n)+6.09*cos(theta(n));
```

```
            TVLA2x=TVLA1x+5.53*sin(theta(n)+theta2)+1.79*cos(theta(n)+theta2);
            TVLA2y=TVLA1y+5.53*cos(theta(n)+theta2)-1.79*sin(theta(n)+theta2);
            TVLA3x=TVLA2x-3.58*cos(theta(n)+theta2);
            TVLA3y=TVLA2y+3.58*sin(theta(n)+theta2);
            fm1131x=TVLA3x-2.38*cos(theta(n)+theta2);
            fm1131y=TVLA3y+2.38*sin(theta(n)+theta2);
            fm1132x=fm1131x+14.51*sin(theta(n)+theta2);
            fm1132y=fm1131y+14.51*cos(theta(n)+theta2);
            fm1133x=fm1132x+8.33*cos(theta(n)+theta2);
            fm1133y=fm1132y-8.33*sin(theta(n)+theta2);
            fm1134x=fm1133x-14.51*sin(theta(n)+theta2);
            fm1134y=fm1133y-14.51*cos(theta(n)+theta2);
            set(tvla,...
                'xdata',[TVLA1x TVLA2x TVLA3x TVLA1x],...
                'ydata',[TVLA1y TVLA2y TVLA3y TVLA1y]);
            set(fm113,...
                'xdata',[fm1131x fm1132x fm1133x fm1134x fm1131x],...
                'ydata',[fm1131y fm1132y fm1133y fm1134y fm1131y]);
    %         set(progHndl,...
    %             'String',['L1=' num2str(L1) ...
    %                 '; alpha_o=' num2str(alpha_o*180/pi) ...
    %                 '; gamma=' num2str(gamma*180/pi)]);
            drawnow;
            while(x~=1)
                x=get(gca,'Userdata');
                switch x
                    case -1
                        set(pauseHndl,'Enable','off');
                        set(unpauseHndl,'Enable','on');
                        set(gca,'Userdata',-2);
                    case 0
                        set(unpauseHndl,'Enable','off');
                        set(pauseHndl,'Enable','on');
                        set(gca,'Userdata',1);
                    case -3
                        set(gca,'Userdata',1);
                        n=L;
                    otherwise
                        pause(0.01);
                end;

            end;
            n=n+skipnum;
        end;
        set([runHndl closeHndl],'Enable','on');
        set([timeHndl pauseHndl unpauseHndl stopHndl],'Enable','off');
    end;
```

trkslpcalc.m

```
1   %  TRKSLPCALC - Track Slip Calculator
2   %    Plots the vehicle paths as calculated by YAW/SPEED
3   %  and L2TRK/R2TRK.  Allows user to change the time and
4   %  adjust the track slip of each track individually.  Will
5   %  then replot the data.  Allows writing and loading of
6   %  track slip data files.
7   %
8   %  Ex.
9   %  trkslpcalc(t,speed,yaw,l2trk,r2trk);
10
11     function trkslpcalc(t,speed,yaw,l2trk,r2trk,action)
12
13     if nargin<5
14         error('You must input the five variables!')
15         return;
16     elseif nargin<6
17         action='initialize';
```

```
18      end;
19
20      if strcmp(action,'initialize');
21
22          figNumber=figure(...
23              'Name','M113 Trackslip Determination Program',...
24              'NumberTitle','off',...
25              'BackingStore','off',...
26              'Visible','off',...
27              'Units','normalized',...
28              'Position',[0.1 0.1 0.8 0.8]);
29          colordef(figNumber,'black');
30          axHndl = axes('Units','normalized',...
31              'Position',[0.05 0.2 0.75 0.75],...
32              'Visible','off');
33          textHndl=text(0,0,'Press the "Plot" button to plot the vehicle paths.',...
34              'HorizontalAlignment','center');
35          axis([-1 1 -1 1]);
36
37          %===================================
38          % Information for button
39          labelColor=[0.8 0.8 0.8];
40          yInitPos=0.50;
41          xPos=0.85;
42          btnLen=0.10;
43          btnWid=0.10;
44          txtWid=0.02;
45          sliderLen=0.2;
46          txtLen=0.05;
47          spacing=0.05;
48
49          %===================================
50          % The CONSOLE frame #1
51          frmBorder=0.02;
52          yPos=0.5+1.5*spacing;
53          frmPos=[xPos-frmBorder yPos btnLen+2*frmBorder spacing+2*btnWid+2*frmBorder];
54          h=uicontrol('Style','frame',...
55              'Units','normalized',...
56              'Position',frmPos,...
57              'BackgroundColor',[0.5 0.5 0.5]);
58
59          %===================================
60          % The CONSOLE frame #2
61          frmBorder=0.02;
62          yPos=0.045-frmBorder;
63          frmPos=[0.1-frmBorder yPos spacing+txtLen+sliderLen+2*frmBorder
    spacing+2*txtWid+2*frmBorder];
64          h=uicontrol('Style','frame',...
65              'Units','normalized',...
66              'Position',frmPos,...
67              'BackgroundColor',[0.5 0.5 0.5]);
68
69          %===================================
70          % The CONSOLE frame #3
71          frmBorder=0.02;
72          yPos=0.045-frmBorder;
73          frmPos=[0.5-frmBorder yPos .33 spacing+2*txtWid+2*frmBorder];
74          h=uicontrol('Style','frame',...
75              'Units','normalized',...
76              'Position',frmPos,...
77              'BackgroundColor',[0.5 0.5 0.5]);
78
79          %===================================
80          % The CONSOLE frame #4
81          frmBorder=0.02;
82          yPos=0.5-2.5*spacing-2*frmBorder-2*btnWid;
83          frmPos=[xPos-frmBorder yPos btnLen+2*frmBorder spacing+2*btnWid+2*frmBorder];
84          h=uicontrol('Style','frame',...
85              'Units','normalized',...
86              'Position',frmPos,...
87              'BackgroundColor',[0.5 0.5 0.5]);
```

```
88
89          %==================================
90          % The TIME slider #1
91          btnNumber=1;
92          cmdStr='time1';
93          callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''ts1'');';
94
95          %Generic button information
96          btnPos=[.1 0.04+txtWid+spacing sliderLen txtWid];
97          time1Hndl=uicontrol(...
98              'Style','slider',...
99              'Units','normalized',...
100             'Position',btnPos,...
101             'SliderStep',[.001 .05],...
102             'Max',max(t),...
103             'Min',min(t),...
104             'Value',t(1),...
105             'Enable','off',...
106             'Callback',callbackstr);
107
108         %==================================
109         % The TIME slider #2
110         btnNumber=2;
111         cmdStr='time2';
112         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''ts2'');';
113
114         %Generic button information
115         btnPos=[.1 0.04 sliderLen txtWid];
116         time2Hndl=uicontrol(...
117             'Style','slider',...
118             'Units','normalized',...
119             'Position',btnPos,...
120             'SliderStep',[.001 .05],...
121             'Max',max(t),...
122             'Min',min(t),...
123             'Value',max(t),...
124             'Enable','off',...
125             'Callback',callbackstr);
126
127         %==================================
128         % The TIME edit #1
129         btnNumber=2;
130         cmdStr='time1';
131         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''te1'');';
132
133         %Generic button information
134         btnPos=[.1+sliderLen+spacing 0.04+txtWid+spacing txtLen txtWid];
135         time1edHndl=uicontrol(...
136             'Style','edit',...
137             'Units','normalized',...
138             'Position',btnPos,...
139             'String',num2str(t(1)),...
140             'Enable','off',...
141             'Callback',callbackstr);
142
143         %==================================
144         % The TIME edit #2
145         btnNumber=2;
146         cmdStr='time2';
147         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''te2'');';
148
149         %Generic button information
150         btnPos=[.1+sliderLen+spacing 0.04 txtLen txtWid];
151         time2edHndl=uicontrol(...
152             'Style','edit',...
153             'Units','normalized',...
154             'Position',btnPos,...
155             'String',num2str(max(t)),...
156             'Enable','off',...
157             'Callback',callbackstr);
158
```

```
159         %==================================
160         % The Plot button
161         btnNumber=3;
162         yPos=0.5+frmBorder+2.5*spacing+btnWid;
163         labelStr='Plot';
164         cmdStr='plot';
165         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''plot'');';
166
167         %Generic button information
168         btnPos=[xPos yPos btnLen btnWid];
169         plotHndl=uicontrol(...
170             'Style','pushbutton',...
171             'Units','normalized',...
172             'Position',btnPos,...
173             'String',labelStr,...
174             'Interruptible','on',...
175             'Callback',callbackstr);
176
177         %==================================
178         % The Close button
179         btnNumber=4;
180         yPos=0.5+frmBorder+1.5*spacing;
181         labelStr='Close';
182         cmdStr='close';
183         callbackstr='close(gcf);';
184
185         %Generic button information
186         btnPos=[xPos yPos btnLen btnWid];
187         closeHndl=uicontrol(...
188             'Style','pushbutton',...
189             'Units','normalized',...
190             'Position',btnPos,...
191             'String',labelStr,...
192             'Interruptible','on',...
193             'Callback',callbackstr);
194
195         %==================================
196         % The Load button
197         btnNumber=8;
198         yPos=0.5-frmBorder-1.5*spacing-btnWid;
199         labelStr='Load Trackslip File';
200         cmdStr='load';
201         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''load'');';
202
203         %Generic button information
204         btnPos=[xPos yPos btnLen btnWid];
205         loadHndl=uicontrol(...
206             'Style','pushbutton',...
207             'Units','normalized',...
208             'Position',btnPos,...
209             'String',labelStr,...
210             'Interruptible','on',...
211             'Enable','off',...
212             'Callback',callbackstr);
213
214         %==================================
215         % The Write button
216         btnNumber=9;
217         yPos=0.5-frmBorder-2.5*spacing-2*btnWid;
218         labelStr='Write Trackslip File';
219         cmdStr='write';
220         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''write'');';
221
222         %Generic button information
223         btnPos=[xPos yPos btnLen btnWid];
224         writeHndl=uicontrol(...
225             'Style','pushbutton',...
226             'Units','normalized',...
227             'Position',btnPos,...
228             'String',labelStr,...
229             'Interruptible','on',...
```

```
230             'Enable','off',...
231             'Callback',callbackstr);
232 
233     %     %==================================
234     %     % The Track Slip Left edit
235     %     btnNumber=5;
236     %     cmdStr='time2';
237     %
238     %     %Generic button information
239     %     btnPos=[.515 0.04+txtWid+spacing txtLen txtWid];
240     %     tslHndl=uicontrol(...
241     %         'Style','edit',...
242     %         'Units','normalized',...
243     %         'Position',btnPos,...
244     %         'String','1',...
245     %         'Enable','off');
246     %
247     %     %==================================
248     %     % The Track Slip Right edit
249     %     btnNumber=6;
250     %     cmdStr='time2';
251     %
252     %     %Generic button information
253     %     btnPos=[.62 0.04+txtWid+spacing txtLen txtWid];
254     %     tsrHndl=uicontrol(...
255     %         'Style','edit',...
256     %         'Units','normalized',...
257     %         'Position',btnPos,...
258     %         'String','1',...
259     %         'Enable','off');
260     %
261     %     %==================================
262     %     % The Set button
263     %     btnNumber=7;
264     %     labelStr='Set';
265     %     cmdStr='Set';
266     %     callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''set'');';
267     %
268     %     %Generic button information
269     %     btnPos=[.568 .04 btnLen/2 btnWid/3];
270     %     setHndl=uicontrol(...
271     %         'Style','pushbutton',...
272     %         'Units','normalized',...
273     %         'Position',btnPos,...
274     %         'String',labelStr,...
275     %         'Interruptible','on',...
276     %         'Enable','off',...
277     %         'Callback',callbackstr);
278 
279         %==================================
280         % The Pivot Radius button
281         btnNumber=9;
282         yPos=0.5-txtWid/2;
283         labelStr='50';
284         cmdStr='prad';
285         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''prad'');';
286 
287         %Generic button information
288         btnPos=[xPos+btnLen/2 yPos btnLen/2 txtWid];
289         pradHndl=uicontrol(...
290             'Style','edit',...
291             'Units','normalized',...
292             'Position',btnPos,...
293             'String',labelStr,...
294             'Interruptible','on',...
295             'Enable','off',...
296             'Userdata',50,...
297             'Callback',callbackstr);
298 
299         %==================================
300         % The tscalc button
```

```
301         btnNumber=10;
302         yPos=0.1;
303         labelStr='Trackslip Calculator';
304         cmdStr='tscalc';
305         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''tscalc'');';
306 
307         %Generic button information
308         btnPos=[xPos yPos btnLen txtWid];
309         tscalcHndl=uicontrol(...
310             'Style','pushbutton',...
311             'Units','normalized',...
312             'Position',btnPos,...
313             'String',labelStr,...
314             'Interruptible','on',...
315             'Enable','off',...
316             'Callback',callbackstr);
317 
318         %==================================
319         % The tsplot button
320         btnNumber=11;
321         yPos=0.07;
322         labelStr='Trackslip Plotter';
323         cmdStr='tsplot';
324         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''tsplot'');';
325 
326         %Generic button information
327         btnPos=[xPos yPos btnLen txtWid];
328         tsplotHndl=uicontrol(...
329             'Style','pushbutton',...
330             'Units','normalized',...
331             'Position',btnPos,...
332             'String',labelStr,...
333             'Interruptible','on',...
334             'Enable','off',...
335             'Callback',callbackstr);
336 
337         %==================================
338         % The Curve Write button
339         btnNumber=12;
340         yPos=0.04;
341         labelStr='Curve Writer';
342         cmdStr='cwrite';
343         callbackstr='crvwrite;';
344 
345         %Generic button information
346         btnPos=[xPos yPos btnLen txtWid];
347         cwriteHndl=uicontrol(...
348             'Style','pushbutton',...
349             'Units','normalized',...
350             'Position',btnPos,...
351             'String',labelStr,...
352             'Interruptible','on',...
353             'Enable','off',...
354             'Callback',callbackstr);
355 
356         %==================================
357         % The TIME slider #3
358         btnNumber=13;
359         cmdStr='time3';
360         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''ts3'');';
361 
362         %Generic button information
363         btnPos=[.495 0.07 sliderLen txtWid];
364         time3Hndl=uicontrol(...
365             'Style','slider',...
366             'Units','normalized',...
367             'Position',btnPos,...
368             'SliderStep',[.001 .05],...
369             'Max',max(t),...
370             'Min',min(t),...
371             'Value',t(round(.5.*max(size(t)))),...
```

```
372             'Enable','off',...
373             'Callback',callbackstr);
374
375         %==================================
376         % The TIME edit #3
377         btnNumber=14;
378         cmdStr='time3';
379         callbackstr='trkslpcalc(t,speed,yaw,l2trk,r2trk,''te3'');';
380
381         %Generic button information
382         btnPos=[.495+sliderLen+spacing 0.07 txtLen txtWid];
383         time3edHndl=uicontrol(...
384             'Style','edit',...
385             'Units','normalized',...
386             'Position',btnPos,...
387             'String',num2str(t(round(.5.*max(size(t))))),...
388             'Enable','off',...
389             'Callback',callbackstr);
390
391         %==================================
392         % Text
393         btnPos=[xPos-.5*btnLen/2 0.5-.0075 .7*btnLen .015];
394         uicontrol(...
395             'Style','text',...
396             'Units','normalized',...
397             'Position',btnPos,...
398             'String','Pivot Radius (in.)',...
399             'BackgroundColor',[.5 .5 .5]);
400
401         btnPos=[.09 .132 .04 .015];
402         uicontrol(...
403             'Style','text',...
404             'Units','normalized',...
405             'Position',btnPos,...
406             'String',t(1),...
407             'BackgroundColor',[.5 .5 .5]);
408
409         btnPos=[.27 .132 .04 .015];
410         uicontrol(...
411             'Style','text',...
412             'Units','normalized',...
413             'Position',btnPos,...
414             'String',num2str(t(max(size(t))-2)),...
415             'BackgroundColor',[.5 .5 .5]);
416
417         btnPos=[.09 .062 .04 .015];
418         uicontrol(...
419             'Style','text',...
420             'Units','normalized',...
421             'Position',btnPos,...
422             'String',num2str(t(3)),...
423             'BackgroundColor',[.5 .5 .5]);
424
425         btnPos=[.27 .062 .04 .015];
426         uicontrol(...
427             'Style','text',...
428             'Units','normalized',...
429             'Position',btnPos,...
430             'String',num2str(t(max(size(t)))),...
431             'BackgroundColor',[.5 .5 .5]);
432
433         btnPos=[.15 .132 .1 .015];
434         uicontrol(...
435             'Style','text',...
436             'Units','normalized',...
437             'Position',btnPos,...
438             'String','Start Time Marker',...
439             'ForegroundColor','g',...
440             'BackgroundColor',[.5 .5 .5]);
441
442         btnPos=[.15 .062 .1 .015];
```

```
443         uicontrol(...
444             'Style','text',...
445             'Units','normalized',...
446             'Position',btnPos,...
447             'String','End Time Marker',...
448             'ForegroundColor','r',...
449             'BackgroundColor',[.5 .5 .5]);
450 
451         btnPos=[.355 .132 .04 .015];
452         uicontrol(...
453             'Style','text',...
454             'Units','normalized',...
455             'Position',btnPos,...
456             'String','Value',...
457             'BackgroundColor',[.5 .5 .5]);
458 
459         btnPos=[.355 .062 .04 .015];
460         uicontrol(...
461             'Style','text',...
462             'Units','normalized',...
463             'Position',btnPos,...
464             'String','Value',...
465             'BackgroundColor',[.5 .5 .5]);
466 
467         btnPos=[.484 .092 .04 .015];
468         uicontrol(...
469             'Style','text',...
470             'Units','normalized',...
471             'Position',btnPos,...
472             'String',num2str(t(2)),...
473             'BackgroundColor',[.5 .5 .5]);
474 
475         btnPos=[.664 .092 .04 .015];
476         uicontrol(...
477             'Style','text',...
478             'Units','normalized',...
479             'Position',btnPos,...
480             'String',num2str(t(max(size(t))-1)),...
481             'BackgroundColor',[.5 .5 .5]);
482 
483         btnPos=[.749 .092 .04 .015];
484         uicontrol(...
485             'Style','text',...
486             'Units','normalized',...
487             'Position',btnPos,...
488             'String','Value',...
489             'BackgroundColor',[.5 .5 .5]);
490 
491         btnPos=[.544 .092 .1 .015];
492         uicontrol(...
493             'Style','text',...
494             'Units','normalized',...
495             'Position',btnPos,...
496             'String','S-Curve Center Marker',...
497             'ForegroundColor',[1 0.7 0],...
498             'BackgroundColor',[.5 .5 .5]);
499 
500         trkslpl = ones(size(t));
501         trkslpr = ones(size(t));
502 
503         set(figNumber,'Visible','on');
504         set(closeHndl,'UserData',[trkslpl trkslpr]);
505         set(axHndl,'UserData',[figNumber axHndl textHndl time1Hndl ...
506                 time2Hndl...
507                 time1edHndl...
508                 time2edHndl...
509                 plotHndl...
510                 closeHndl...
511                 loadHndl...
512                 writeHndl,...
513                 pradHndl,...
```

```
514                 tscalcHndl,...
515                 tsplotHndl,...
516                 cwriteHndl,...
517                 time3Hndl,...
518                 time3edHndl]);
519
520     %=========================================================================
    ==
521     %=========================================================================
    ==
522
523     elseif strcmp(action,'plot')
524         Hndllist = get(gca,'Userdata');
525         figNumber = Hndllist(1);
526         axHndl = Hndllist(2);
527         textHndl = Hndllist(3);
528         time1Hndl = Hndllist(4);
529         time2Hndl = Hndllist(5);
530         time1edHndl = Hndllist(6);
531         time2edHndl = Hndllist(7);
532         plotHndl = Hndllist(8);
533         closeHndl = Hndllist(9);
534     %     ts1Hndl = Hndllist(10);
535     %     tsrHndl = Hndllist(11);
536     %     setHndl = Hndllist(12);
537         loadHndl = Hndllist(10);
538         writeHndl = Hndllist(11);
539         pradHndl = Hndllist(12);
540         tscalcHndl = Hndllist(13);
541         tsplotHndl = Hndllist(14);
542         cwriteHndl = Hndllist(15);
543         time3Hndl = Hndllist(16);
544         time3edHndl = Hndllist(17);
545
546         prad=get(pradHndl,'Userdata');
547
548         set(textHndl,'Visible','off');
549         set(axHndl,'Visible','on');
550         set(plotHndl,'String','Regenerate',...
551             'Callback','trkslpcalc(t,speed,yaw,l2trk,r2trk,''regen'');');
552         set(Hndllist(4:7),'Enable','on');
553         set(Hndllist(10:17),'Enable','on');
554
555         trkslp = get(closeHndl,'Userdata');
556         trkslpl=trkslp(:,1);
557         trkslpr=trkslp(:,2);
558         %============================
559         %Calculations
560
561         l2trk=l2trk./trkslpl;
562         r2trk=r2trk./trkslpr;
563
564         speed=88.*speed./60;
565         l2trk=88.*l2trk./60;
566         r2trk=88.*r2trk./60;
567
568         speed=(63/12).*pi.*yaw./180+speed;  %Corrects for 5th wheel offset.
569
570         speed2=(l2trk+r2trk)./2;
571
572         theta=cumtrapz(t,yaw);
573         theta2=cumtrapz(t,(6./prad).*(l2trk-r2trk));
574         theta=pi*theta./180;
575
576         fpos=cumtrapz(t,speed);
577         fpos2=cumtrapz(t,speed2);
578
579
580         fpos=[fpos(1);diff(fpos)];
581         fpos2=[fpos2(1);diff(fpos2)];
582
```

```
583
584         cgx=cumsum(fpos.*sin(theta));
585         cgy=cumsum(fpos.*cos(theta));
586         cgx2=cumsum(fpos2.*sin(theta2));
587         cgy2=cumsum(fpos2.*cos(theta2));
588
589         set(axHndl,'XLim',[-10+min(min([cgx cgx2])) 10+max(max([cgx cgx2]))],...
590                 'YLim',[-10+min(min([cgy cgy2])) 10+max(max([cgy cgy2]))]);
591
592         val1 = get(time1Hndl,'Value');
593         val2 = get(time2Hndl,'Value');
594         val3 = get(time3Hndl,'Value');
595
596         indx1=round(mean(find(t==val1)));
597         indx2=round(mean(find(t==val2)));
598         indx3=round(mean(find(t==val3)));
599
600
601         body=line('color','b',...
602             'LineStyle','-',...
603             'xdata',cgx,'ydata',cgy);
604         body2=line('color','w',...
605             'LineStyle','-',...
606             'xdata',cgx2,'ydata',cgy2);
607         head=line('color','g',...
608             'Marker','*',...
609             'markersize',10,...
610             'xdata',cgx(indx1:indx1+1),'ydata',cgy(indx1:indx1+1));
611         head2=line('color','g',...
612             'Marker','*',...
613             'markersize',10,...
614             'xdata',cgx2(indx1:indx1+1),'ydata',cgy2(indx1:indx1+1));
615         mid=line('color',[1 .7 0],...
616             'Marker','*',...
617             'Markersize',10,...
618             'xdata',cgx(indx3:indx3+1),'ydata',cgy(indx3:indx3+1));
619         mid2=line('color',[1 .7 0],...
620             'Marker','*',...
621             'Markersize',10,...
622             'xdata',cgx2(indx3:indx3+1),'ydata',cgy2(indx3:indx3+1));
623         tail=line('color','r',...
624             'Marker','*',...
625             'markersize',10,...
626             'xdata',cgx(indx2-1:indx2),'ydata',cgy(indx2-1:indx2));
627         tail2=line('color','r',...
628             'Marker','*',...
629             'markersize',10,...
630             'xdata',cgx2(indx2-1:indx2),'ydata',cgy2(indx2-1:indx2));
631
632         set(plotHndl,'Userdata',[body body2 head head2 tail tail2 mid mid2]);
633         set(time1Hndl,'Userdata',[cgx cgy cgx2 cgy2]);
634         set(cwriteHndl,'Userdata',[t cgx2 cgy2 trkslpl trkslpr speed yaw]);
635
636
637     %=============================================================================
    ==
638     %=============================================================================
    ==
639
640     elseif strcmp(action,'ts1')
641         Hndllist = get(gca,'Userdata');
642         figNumber = Hndllist(1);
643         axHndl = Hndllist(2);
644         textHndl = Hndllist(3);
645         time1Hndl = Hndllist(4);
646         time2Hndl = Hndllist(5);
647         time1edHndl = Hndllist(6);
648         time2edHndl = Hndllist(7);
649         plotHndl = Hndllist(8);
650         closeHndl = Hndllist(9);
651         %     ts1Hndl = Hndllist(10);
```

```
652         %     tsrHndl = Hndllist(11);
653         %     setHndl = Hndllist(12);
654         time3Hndl = Hndllist(16);
655         time3edHndl = Hndllist(17);
656 
657         linelist = get(plotHndl,'Userdata');
658         body = linelist(1);
659         body2 = linelist(2);
660         head = linelist(3);
661         head2 = linelist(4);
662         tail = linelist(5);
663         tail2 = linelist(6);
664         mid = linelist(7);
665         mid2 = linelist(8);
666 
667         cglist = get(time1Hndl,'Userdata');
668         cgx = cglist(:,1);
669         cgy = cglist(:,2);
670         cgx2 = cglist(:,3);
671         cgy2 = cglist(:,4);
672 
673         val=get(time1Hndl,'Value');
674         indx=round(mean(find(t>=val-0.05 & t<=val+0.05)));
675         if val>=get(time2Hndl,'Value')
676             set(time1Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-2));
677             set(time1edHndl,'String',num2str(get(time1Hndl,'Value')));
678             set(time3Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-1));
679             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
680         elseif val<t(4)
681             set(time1edHndl,'String',num2str(min(t)));
682             set(time1Hndl,'Value',min(t));
683         else
684             set(time1edHndl,'String',num2str(t(indx)));
685             set(time1Hndl,'Value',t(indx));
686         end
687         if get(time3Hndl,'Value')<=get(time1Hndl,'Value')
688             set(time3Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+1));
689             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
690         end
691         indx1=round(mean(find(t==get(time1Hndl,'Value'))));
692         set(head,'XData',cgx(indx1:indx1+1),'YData',cgy(indx1:indx1+1));
693         set(head2,'XData',cgx2(indx1:indx1+1),'YData',cgy2(indx1:indx1+1));
694         indx3=round(mean(find(t==get(time3Hndl,'Value'))));
695         set(mid,'XData',cgx(indx3:indx3+1),'YData',cgy(indx3:indx3+1));
696         set(mid2,'XData',cgx2(indx3:indx3+1),'YData',cgy2(indx3:indx3+1));
697 
698     %=================================================================================
    ==
699     %=================================================================================
    ==
700 
701     elseif strcmp(action,'ts2')
702         Hndllist = get(gca,'Userdata');
703         figNumber = Hndllist(1);
704         axHndl = Hndllist(2);
705         textHndl = Hndllist(3);
706         time1Hndl = Hndllist(4);
707         time2Hndl = Hndllist(5);
708         time1edHndl = Hndllist(6);
709         time2edHndl = Hndllist(7);
710         plotHndl = Hndllist(8);
711         closeHndl = Hndllist(9);
712     %     ts1Hndl = Hndllist(10);
713     %     tsrHndl = Hndllist(11);
714     %     setHndl = Hndllist(12);
715         time3Hndl = Hndllist(16);
716         time3edHndl = Hndllist(17);
717 
718         linelist = get(plotHndl,'Userdata');
719         body = linelist(1);
720         body2 = linelist(2);
```

```
721         head = linelist(3);
722         head2 = linelist(4);
723         tail = linelist(5);
724         tail2 = linelist(6);
725         mid = linelist(7);
726         mid2 = linelist(8);
727 
728         cglist = get(time1Hndl,'Userdata');
729         cgx = cglist(:,1);
730         cgy = cglist(:,2);
731         cgx2 = cglist(:,3);
732         cgy2 = cglist(:,4);
733 
734         val=get(time2Hndl,'Value');
735         indx=round(mean(find(t>=val-0.05 & t<=val+0.05)));
736         if val<=get(time1Hndl,'Value')
737             set(time2Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+2));
738             set(time2edHndl,'String',num2str(get(time2Hndl,'Value')));
739             set(time3Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+1));
740             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
741         elseif val>t(max(size(t))-4)
742             set(time2edHndl,'String',num2str(max(t)));
743             set(time2Hndl,'Value',max(t));
744         else
745             set(time2edHndl,'String',num2str(t(indx)));
746             set(time2Hndl,'Value',t(indx));
747         end
748         if get(time3Hndl,'Value')>=get(time2Hndl,'Value')
749             set(time3Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-1));
750             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
751         end
752         indx1=round(mean(find(t==get(time2Hndl,'Value'))));
753         set(tail,'XData',cgx(indx1-1:indx1),'YData',cgy(indx1-1:indx1));
754         set(tail2,'XData',cgx2(indx1-1:indx1),'YData',cgy2(indx1-1:indx1));
755         indx3=round(mean(find(t==get(time3Hndl,'Value'))));
756         set(mid,'XData',cgx(indx3:indx3+1),'YData',cgy(indx3:indx3+1));
757         set(mid2,'XData',cgx2(indx3:indx3+1),'YData',cgy2(indx3:indx3+1));
758 
759     %=================================================================================
    ==
760     %=================================================================================
    ==
761 
762     elseif strcmp(action,'ts3')
763         Hndllist = get(gca,'Userdata');
764         figNumber = Hndllist(1);
765         axHndl = Hndllist(2);
766         textHndl = Hndllist(3);
767         time1Hndl = Hndllist(4);
768         time2Hndl = Hndllist(5);
769         time1edHndl = Hndllist(6);
770         time2edHndl = Hndllist(7);
771         plotHndl = Hndllist(8);
772         closeHndl = Hndllist(9);
773     %     ts1Hndl = Hndllist(10);
774     %     tsrHndl = Hndllist(11);
775     %     setHndl = Hndllist(12);
776         time3Hndl = Hndllist(16);
777         time3edHndl = Hndllist(17);
778 
779         linelist = get(plotHndl,'Userdata');
780         body = linelist(1);
781         body2 = linelist(2);
782         head = linelist(3);
783         head2 = linelist(4);
784         tail = linelist(5);
785         tail2 = linelist(6);
786         mid = linelist(7);
787         mid2 = linelist(8);
788 
789         cglist = get(time1Hndl,'Userdata');
```

```
790         cgx = cglist(:,1);
791         cgy = cglist(:,2);
792         cgx2 = cglist(:,3);
793         cgy2 = cglist(:,4);
794 
795         val=get(time3Hndl,'Value');
796         indx=round(mean(find(t>=val-0.05 & t<=val+0.05)));
797         if val<=get(time1Hndl,'Value')
798             set(time3Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+1));
799             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
800         elseif val>=get(time2Hndl,'Value')
801             set(time3Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-1));
802             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
803         else
804             set(time3edHndl,'String',num2str(t(indx)));
805             set(time3Hndl,'Value',t(indx));
806         end
807         indx3=round(mean(find(t==get(time3Hndl,'Value'))));
808         set(mid,'XData',cgx(indx3:indx3+1),'YData',cgy(indx3:indx3+1));
809         set(mid2,'XData',cgx2(indx3:indx3+1),'YData',cgy2(indx3:indx3+1));
810 
811     %=============================================================================
    ==
812     %=============================================================================
    ==
813 
814     elseif strcmp(action,'te1')
815         Hndllist = get(gca,'Userdata');
816         figNumber = Hndllist(1);
817         axHndl = Hndllist(2);
818         textHndl = Hndllist(3);
819         time1Hndl = Hndllist(4);
820         time2Hndl = Hndllist(5);
821         time1edHndl = Hndllist(6);
822         time2edHndl = Hndllist(7);
823         plotHndl = Hndllist(8);
824         closeHndl = Hndllist(9);
825         time3Hndl = Hndllist(16);
826         time3edHndl = Hndllist(17);
827 
828         linelist = get(plotHndl,'Userdata');
829         body = linelist(1);
830         body2 = linelist(2);
831         head = linelist(3);
832         head2 = linelist(4);
833         tail = linelist(5);
834         tail2 = linelist(6);
835         mid = linelist(7);
836         mid2 = linelist(8);
837 
838         cglist = get(time1Hndl,'Userdata');
839         cgx = cglist(:,1);
840         cgy = cglist(:,2);
841         cgx2 = cglist(:,3);
842         cgy2 = cglist(:,4);
843 
844         val=str2double(get(time1edHndl,'String'));
845 
846         if val>=get(time2Hndl,'Value')
847             set(time1Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-2));
848             set(time1edHndl,'String',num2str(get(time1Hndl,'Value')));
849             set(time3Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-1));
850             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
851         elseif val<t(4)
852             set(time1edHndl,'String',num2str(min(t)));
853             set(time1Hndl,'Value',min(t));
854         else
855             indx=round(mean(find(t>=val-0.05 & t<=val+0.05)));
856             set(time1edHndl,'String',num2str(t(indx)));
857             set(time1Hndl,'Value',t(indx));
858         end
```

```
859         if get(time3Hndl,'Value')<=get(time1Hndl,'Value')
860             set(time3Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+1));
861             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
862         end
863         indx3=round(mean(find(t==get(time3Hndl,'Value'))));
864         set(mid,'XData',cgx(indx3:indx3+1),'YData',cgy(indx3:indx3+1));
865         set(mid2,'XData',cgx2(indx3:indx3+1),'YData',cgy2(indx3:indx3+1));
866         indx1=round(mean(find(t==get(time1Hndl,'Value'))));
867         set(head,'XData',cgx(indx1:indx1+1),'YData',cgy(indx1:indx1+1));
868         set(head2,'XData',cgx2(indx1:indx1+1),'YData',cgy2(indx1:indx1+1));
869
870     %=============================================================================
    ==
871     %=============================================================================
    ==
872
873     elseif strcmp(action,'te2')
874         Hndllist = get(gca,'Userdata');
875         figNumber = Hndllist(1);
876         axHndl = Hndllist(2);
877         textHndl = Hndllist(3);
878         time1Hndl = Hndllist(4);
879         time2Hndl = Hndllist(5);
880         time1edHndl = Hndllist(6);
881         time2edHndl = Hndllist(7);
882         plotHndl = Hndllist(8);
883         closeHndl = Hndllist(9);
884         time3Hndl = Hndllist(16);
885         time3edHndl = Hndllist(17);
886
887         linelist = get(plotHndl,'Userdata');
888         body = linelist(1);
889         body2 = linelist(2);
890         head = linelist(3);
891         head2 = linelist(4);
892         tail = linelist(5);
893         tail2 = linelist(6);
894         mid = linelist(7);
895         mid2 = linelist(8);
896
897         cglist = get(time1Hndl,'Userdata');
898         cgx = cglist(:,1);
899         cgy = cglist(:,2);
900         cgx2 = cglist(:,3);
901         cgy2 = cglist(:,4);
902
903         val=str2double(get(time2edHndl,'String'));
904
905         if val<=get(time1Hndl,'Value')
906             set(time2Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+2));
907             set(time2edHndl,'String',num2str(get(time2Hndl,'Value')));
908             set(time3Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+1));
909             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
910         elseif val>t(max(size(t))-4)
911             set(time2edHndl,'String',num2str(max(t)));
912             set(time2Hndl,'Value',max(t));
913         else
914             indx=round(mean(find(t>=val-0.05 & t<=val+0.05)));
915             set(time2edHndl,'String',num2str(t(indx)));
916             set(time2Hndl,'Value',t(indx));
917         end
918         if get(time3Hndl,'Value')>=get(time2Hndl,'Value')
919             set(time3Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-1));
920             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
921         end
922         indx3=round(mean(find(t==get(time3Hndl,'Value'))));
923         set(mid,'XData',cgx(indx3:indx3+1),'YData',cgy(indx3:indx3+1));
924         set(mid2,'XData',cgx2(indx3:indx3+1),'YData',cgy2(indx3:indx3+1));
925
926         indx1=round(mean(find(t==get(time2Hndl,'Value'))));
927         set(tail,'XData',cgx(indx1-1:indx1),'YData',cgy(indx1-1:indx1));
```

```
928         set(tail2,'XData',cgx2(indx-1:indx1),'YData',cgy2(indx1-1:indx1));
929
930     %=============================================================================
    ==
931     %=============================================================================
    ==
932
933     elseif strcmp(action,'te3')
934         Hndllist = get(gca,'Userdata');
935         figNumber = Hndllist(1);
936         axHndl = Hndllist(2);
937         textHndl = Hndllist(3);
938         time1Hndl = Hndllist(4);
939         time2Hndl = Hndllist(5);
940         time1edHndl = Hndllist(6);
941         time2edHndl = Hndllist(7);
942         plotHndl = Hndllist(8);
943         closeHndl = Hndllist(9);
944         time3Hndl = Hndllist(16);
945         time3edHndl = Hndllist(17);
946
947         linelist = get(plotHndl,'Userdata');
948         body = linelist(1);
949         body2 = linelist(2);
950         head = linelist(3);
951         head2 = linelist(4);
952         tail = linelist(5);
953         tail2 = linelist(6);
954         mid = linelist(7);
955         mid2 = linelist(8);
956
957         cglist = get(time1Hndl,'Userdata');
958         cgx = cglist(:,1);
959         cgy = cglist(:,2);
960         cgx2 = cglist(:,3);
961         cgy2 = cglist(:,4);
962
963         val=str2double(get(time3edHndl,'String'));
964
965         if val<=get(time1Hndl,'Value')
966             set(time3Hndl,'Value',t(round(mean(find(t==get(time1Hndl,'Value'))))+1));
967             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
968         elseif val>=get(time2Hndl,'Value')
969             set(time3Hndl,'Value',t(round(mean(find(t==get(time2Hndl,'Value'))))-1));
970             set(time3edHndl,'String',num2str(get(time3Hndl,'Value')));
971         else
972             indx=round(mean(find(t>=val-0.05 & t<=val+0.05)));
973             set(time3edHndl,'String',num2str(t(indx)));
974             set(time3Hndl,'Value',t(indx));
975         end
976         indx3=round(mean(find(t==get(time3Hndl,'Value'))));
977         set(mid,'XData',cgx(indx3:indx3+1),'YData',cgy(indx3:indx3+1));
978         set(mid2,'XData',cgx2(indx3:indx3+1),'YData',cgy2(indx3:indx3+1));
979
980     %=============================================================================
    ==
981     %=============================================================================
    ==
982
983     elseif strcmp(action,'regen')
984         Hndllist = get(gca,'Userdata');
985         figNumber = Hndllist(1);
986         axHndl = Hndllist(2);
987         textHndl = Hndllist(3);
988         time1Hndl = Hndllist(4);
989         time2Hndl = Hndllist(5);
990         time1edHndl = Hndllist(6);
991         time2edHndl = Hndllist(7);
992         plotHndl = Hndllist(8);
993         closeHndl = Hndllist(9);
994     %     ts1Hndl = Hndllist(10);
```

```
995     %     tsrHndl = Hndllist(11);
996     %     setHndl = Hndllist(12);
997         pradHndl = Hndllist(12);
998         cwriteHndl = Hndllist(15);
999         time3Hndl = Hndllist(16);
1000        time3edHndl = Hndllist(17);
1001
1002
1003        prad=get(pradHndl,'Userdata');
1004
1005        linelist = get(plotHndl,'Userdata');
1006        body = linelist(1);
1007        body2 = linelist(2);
1008        head = linelist(3);
1009        head2 = linelist(4);
1010        tail = linelist(5);
1011        tail2 = linelist(6);
1012        mid = linelist(7);
1013        mid2 = linelist(8);
1014
1015        trkslp = get(closeHndl,'Userdata');
1016        trkslpl=trkslp(:,1);
1017        trkslpr=trkslp(:,2);
1018        %============================
1019        %Calculations
1020
1021        l2trk=l2trk./trkslpl;
1022        r2trk=r2trk./trkslpr;
1023
1024        speed=88.*speed./60;
1025        l2trk=88.*l2trk./60;
1026        r2trk=88.*r2trk./60;
1027
1028        speed=(63/12).*pi.*yaw./180+speed;  %Corrects for 5th wheel offset.
1029
1030        speed2=(l2trk+r2trk)./2;
1031
1032        theta=cumtrapz(t,yaw);
1033        theta2=cumtrapz(t,(6./prad).*(l2trk-r2trk));
1034        theta=pi*theta./180;
1035
1036        fpos=cumtrapz(t,speed);
1037        fpos2=cumtrapz(t,speed2);
1038
1039
1040        fpos=[fpos(1);diff(fpos)];
1041        fpos2=[fpos2(1);diff(fpos2)];
1042
1043
1044        cgx=cumsum(fpos.*sin(theta));
1045        cgy=cumsum(fpos.*cos(theta));
1046        cgx2=cumsum(fpos2.*sin(theta2));
1047        cgy2=cumsum(fpos2.*cos(theta2));
1048
1049        set(axHndl,'XLim',[-10+min(min([cgx cgx2])) 10+max(max([cgx cgx2]))],...
1050                'YLim',[-10+min(min([cgy cgy2])) 10+max(max([cgy cgy2]))]);
1051
1052        val1 = get(time1Hndl,'Value');
1053        val2 = get(time2Hndl,'Value');
1054        val3 = get(time3Hndl,'Value');
1055
1056        indx1=round(mean(find(t==val1)));
1057        indx2=round(mean(find(t==val2)));
1058        indx3=round(mean(find(t==val3)));
1059
1060        set(body,'XData',cgx,'Ydata',cgy);
1061        set(body2,'XData',cgx2,'Ydata',cgy2);
1062        set(head,'XData',cgx(indx1:indx1+1),'Ydata',cgy(indx1:indx1+1));
1063        set(head2,'XData',cgx2(indx1:indx1+1),'Ydata',cgy2(indx1:indx1+1));
1064        set(tail,'XData',cgx(indx2-1:indx2),'Ydata',cgy(indx2-1:indx2));
1065        set(tail2,'XData',cgx2(indx2-1:indx2),'Ydata',cgy2(indx2-1:indx2));
```

```
1066        set(mid,'XData',cgx(indx3-1:indx3),'Ydata',cgy(indx3-1:indx3));
1067        set(mid2,'XData',cgx2(indx3-1:indx3),'Ydata',cgy2(indx3-1:indx3));
1068
1069        set(time1Hndl,'Userdata',[cgx cgy cgx2 cgy2]);
1070        set(cwriteHndl,'Userdata',[t cgx2 cgy2 trkslpl trkslpr speed yaw]);
1071
1072    %================================================================================
    ==
1073    %================================================================================
    ==
1074
1075    elseif strcmp(action,'set')
1076        Hndllist = get(gca,'Userdata');
1077        figNumber = Hndllist(1);
1078        axHndl = Hndllist(2);
1079        textHndl = Hndllist(3);
1080        time1Hndl = Hndllist(4);
1081        time2Hndl = Hndllist(5);
1082        time1edHndl = Hndllist(6);
1083        time2edHndl = Hndllist(7);
1084        plotHndl = Hndllist(8);
1085        closeHndl = Hndllist(9);
1086    %     tslHndl = Hndllist(10);
1087    %     tsrHndl = Hndllist(11);
1088    %     setHndl = Hndllist(12);
1089
1090
1091        trkslp = get(closeHndl,'Userdata');
1092        trkslpl=trkslp(:,1);
1093        trkslpr=trkslp(:,2);
1094
1095        val1=get(time1Hndl,'Value');
1096        val2=get(time2Hndl,'Value');
1097
1098        indx1=round(mean(find(t==val1)));
1099        indx2=round(mean(find(t==val2)));
1100
1101        tslval=str2double(get(tslHndl,'String'));
1102        tsrval=str2double(get(tsrHndl,'String'));
1103
1104
1105        if (isnan(tslval)|tslval<=0) & ~(isnan(tsrval)|tsrval<=0)
1106            tslval=1;
1107            set(tslHndl,'String','1');
1108            lerrNumber=figure(...
1109                'Name','Value Error',...
1110                'NumberTitle','off',...
1111                'BackingStore','off',...
1112                'Visible','on',...
1113                'Units','normalized',...
1114                'Position',[0.44 0.45 0.12 0.1],...
1115                'MenuBar','none');
1116
1117            %==================================
1118            % The Set button
1119            btnNumber=7;
1120            labelStr='Ok';
1121            cmdStr='Close';
1122            callbackstr='close(gcf);';
1123
1124            %Generic button information
1125            btnPos=[.25 .15 .5 .25];
1126            setHndl=uicontrol(...
1127                'Style','pushbutton',...
1128                'Units','normalized',...
1129                'Position',btnPos,...
1130                'String',labelStr,...
1131                'Interruptible','on',...
1132                'Enable','on',...
1133                'Callback',callbackstr);
1134
```

```
1135            %==================================
1136            % Text
1137            btnPos=[.01 .45 .98 .5];
1138            uicontrol(...
1139                'Style','text',...
1140                'Units','normalized',...
1141                'Position',btnPos,...
1142                'String',['''Left Track Slip Value'' is not an '...
1143                'acceptable value.  Value must be a NUMBER greater'...
1144                ' than zero.  Track slip not modified.'],...
1145                'BackgroundColor',[.8 .8 .8]);
1146
1147        elseif (isnan(tsrval)|tsrval<=0) & ~(isnan(tslval)|tslval<=0)
1148            tsrval=1;
1149            set(tsrHndl,'String','1');
1150            lerrNumber=figure(...
1151                'Name','Value Error',...
1152                'NumberTitle','off',...
1153                'BackingStore','off',...
1154                'Visible','on',...
1155                'Units','normalized',...
1156                'Position',[0.44 0.45 0.12 0.1],...
1157                'MenuBar','none');
1158
1159            %==================================
1160            % The OK button
1161            btnNumber=7;
1162            labelStr='Ok';
1163            cmdStr='Close';
1164            callbackstr='close(gcf);';
1165
1166            %Generic button information
1167            btnPos=[.25 .15 .5 .25];
1168            setHndl=uicontrol(...
1169                'Style','pushbutton',...
1170                'Units','normalized',...
1171                'Position',btnPos,...
1172                'String',labelStr,...
1173                'Interruptible','on',...
1174                'Enable','on',...
1175                'Callback',callbackstr);
1176
1177            %==================================
1178            % Text
1179            btnPos=[.01 .45 .98 .5];
1180            uicontrol(...
1181                'Style','text',...
1182                'Units','normalized',...
1183                'Position',btnPos,...
1184                'String',['''Right Track Slip Value'' is not an '...
1185                'acceptable value.  Value must be a NUMBER greater'...
1186                ' than zero.  Track slip not modified.'],...
1187                'BackgroundColor',[.8 .8 .8]);
1188        elseif (isnan(tslval)|tslval<=0) & (isnan(tsrval)|tsrval<=0)
1189            tslval=1;
1190            tsrval=1;
1191            set([tslHndl tsrHndl],'String','1');
1192            lerrNumber=figure(...
1193                'Name','Value Error',...
1194                'NumberTitle','off',...
1195                'BackingStore','off',...
1196                'Visible','on',...
1197                'Units','normalized',...
1198                'Position',[0.44 0.45 0.12 0.1],...
1199                'MenuBar','none');
1200
1201            %==================================
1202            % The Set button
1203            btnNumber=7;
1204            labelStr='Ok';
1205            cmdStr='Close';
```

```
1206            callbackstr='close(gcf);';
1207
1208            %Generic button information
1209            btnPos=[.25 .05 .5 .25];
1210            setHndl=uicontrol(...
1211                'Style','pushbutton',...
1212                'Units','normalized',...
1213                'Position',btnPos,...
1214                'String',labelStr,...
1215                'Interruptible','on',...
1216                'Enable','on',...
1217                'Callback',callbackstr);
1218
1219            %==================================
1220            % Text
1221            btnPos=[.01 .35 .98 .6];
1222            uicontrol(...
1223                'Style','text',...
1224                'Units','normalized',...
1225                'Position',btnPos,...
1226                'String',['Neither the ''Left Track Slip Value'' nor the '...
1227                    '''Right Track Slip Value" is an acceptable value.  ' ...
1228                    'Value must be a NUMBER greater'...
1229                ' than zero.  Track slip not modified.'],...
1230                'BackgroundColor',[.8 .8 .8]);
1231        else
1232            trkslpl(indx1:indx2)=trkslpl(indx1:indx2).*tslval;
1233            trkslpr(indx1:indx2)=trkslpr(indx1:indx2).*tsrval;
1234
1235            set(closeHndl,'Userdata',[trkslpl trkslpr]);
1236            set([tslHndl tsrHndl],'String','1');
1237        end;
1238
1239    %================================================================================
    ==
1240    %================================================================================
    ==
1241
1242    elseif strcmp(action,'load')
1243        Hndllist = get(gca,'Userdata');
1244        figNumber = Hndllist(1);
1245        axHndl = Hndllist(2);
1246        textHndl = Hndllist(3);
1247        time1Hndl = Hndllist(4);
1248        time2Hndl = Hndllist(5);
1249        time1edHndl = Hndllist(6);
1250        time2edHndl = Hndllist(7);
1251        plotHndl = Hndllist(8);
1252        closeHndl = Hndllist(9);
1253    %     tslHndl = Hndllist(10);
1254    %     tsrHndl = Hndllist(11);
1255    %     setHndl = Hndllist(12);
1256        pradHndl = Hndllist(12);
1257
1258        trkslp = get(closeHndl,'Userdata');
1259        trkslpl=trkslp(:,1);
1260        trkslpr=trkslp(:,2);
1261        a=ones(size(trkslpr));
1262        [filename,pathname] = uigetfile('*.asc','Select ASCII file to LOAD');
1263        if filename~=0
1264            m=dlmread([pathname filename],'\t');
1265            if isequal(size(m),size([t trkslp a]))
1266                trkslpl=m(:,2);
1267                trkslpr=m(:,3);
1268                prad=m(1,4);
1269                set(closeHndl,'Userdata',[trkslpl trkslpr]);
1270                set(pradHndl,'Userdata',prad,'String',num2str(prad));
1271            else
1272                lerrNumber=figure(...
1273                    'Name','File Size Error',...
1274                    'NumberTitle','off',...
```

```
1275                    'BackingStore','off',...
1276                    'Visible','on',...
1277                    'Units','normalized',...
1278                    'Position',[0.44 0.45 0.12 0.1],...
1279                    'MenuBar','none');
1280
1281                %==================================
1282                % The OK button
1283                btnNumber=7;
1284                labelStr='Ok';
1285                cmdStr='Close';
1286                callbackstr='close(gcf);';
1287
1288                %Generic button information
1289                btnPos=[.25 .25 .5 .25];
1290                setHndl=uicontrol(...
1291                    'Style','pushbutton',...
1292                    'Units','normalized',...
1293                    'Position',btnPos,...
1294                    'String',labelStr,...
1295                    'Interruptible','on',...
1296                    'Enable','on',...
1297                    'Callback',callbackstr);
1298
1299                %==================================
1300                % Text
1301                btnPos=[.01 .55 .98 .3];
1302                uicontrol(...
1303                    'Style','text',...
1304                    'Units','normalized',...
1305                    'Position',btnPos,...
1306                    'String',['Imported trackslip data is incompatible '...
1307                        'with this vehicle data.'],...
1308                    'BackgroundColor',[.8 .8 .8]);
1309            end;
1310        end;
1311
1312    %=================================================================================
    ==
1313    %=================================================================================
    ==
1314
1315    elseif strcmp(action,'write')
1316        Hndllist = get(gca,'Userdata');
1317        figNumber = Hndllist(1);
1318        axHndl = Hndllist(2);
1319        textHndl = Hndllist(3);
1320        time1Hndl = Hndllist(4);
1321        time2Hndl = Hndllist(5);
1322        time1edHndl = Hndllist(6);
1323        time2edHndl = Hndllist(7);
1324        plotHndl = Hndllist(8);
1325        closeHndl = Hndllist(9);
1326    %     tslHndl = Hndllist(10);
1327    %     tsrHndl = Hndllist(11);
1328    %     setHndl = Hndllist(12);
1329        pradHndl = Hndllist(12);
1330
1331        prad=get(pradHndl,'Userdata');
1332        trkslp = get(closeHndl,'Userdata');
1333        trkslpl=trkslp(:,1);
1334        trkslpr=trkslp(:,2);
1335        pradvect=zeros(size(trkslpr));
1336        pradvect(1)=prad;
1337
1338        [filename,pathname] = uiputfile('*.*','Select ASCII file to LOAD');
1339
1340        if filename~=0
1341            dlmwrite([pathname filename],[t trkslp pradvect],'\t');
1342        end;
1343
```

```
%==========================================================================
==
%==========================================================================
==

elseif strcmp(action,'prad')
    Hndllist = get(gca,'Userdata');
    figNumber = Hndllist(1);
    axHndl = Hndllist(2);
    textHndl = Hndllist(3);
    time1Hndl = Hndllist(4);
    time2Hndl = Hndllist(5);
    time1edHndl = Hndllist(6);
    time2edHndl = Hndllist(7);
    plotHndl = Hndllist(8);
    closeHndl = Hndllist(9);
%     ts1Hndl = Hndllist(10);
%     tsrHndl = Hndllist(11);
%     setHndl = Hndllist(12);
    loadHndl = Hndllist(10);
    writeHndl = Hndllist(11);
    pradHndl = Hndllist(12);

    a=get(pradHndl,'Userdata');
    b=get(pradHndl,'String');
    c=str2double(b);

    if (~isequalwithequalnans(NaN,c)&c>0)
        set(pradHndl,'Userdata',c);
    else
        set(pradHndl,'String',num2str(a));
    end;

%==========================================================================
==
%==========================================================================
==

elseif strcmp(action,'tscalc')
    Hndllist = get(gca,'Userdata');
    figNumber = Hndllist(1);
    axHndl = Hndllist(2);
    textHndl = Hndllist(3);
    time1Hndl = Hndllist(4);
    time2Hndl = Hndllist(5);
    time1edHndl = Hndllist(6);
    time2edHndl = Hndllist(7);
    plotHndl = Hndllist(8);
    closeHndl = Hndllist(9);
%     ts1Hndl = Hndllist(10);
%     tsrHndl = Hndllist(11);
%     setHndl = Hndllist(12);
    loadHndl = Hndllist(10);
    writeHndl = Hndllist(11);
    pradHndl = Hndllist(12);

    a=get(pradHndl,'Userdata');
    tscalc(t,speed,yaw,l2trk,r2trk,a);

%==========================================================================
==
%==========================================================================
==

elseif strcmp(action,'tsplot')
   Hndllist = get(gca,'Userdata');
   closeHndl = Hndllist(9);
   pradHndl = Hndllist(12);

   prad=get(pradHndl,'Userdata');

```

```
1409       trkslp = get(closeHndl,'Userdata');
1410       trkslpl = trkslp(:,1);
1411       trkslpr = trkslp(:,2);
1412
1413       figure('Name','Trackslips',...
1414            'NumberTitle','off',...
1415            'BackingStore','off',...
1416            'Visible','on',...
1417            'Units','normalized',...
1418            'Position',[0.05 0.05 0.9 0.9]);
1419        plot(t,trkslpl,t,trkslpr);
1420        legend('Left Track','Right Track');
1421        title(['Track Speed Ratios (' num2str(prad) ' in. Pivot Radius)']);
1422        xlabel('Time (s)');
1423        ylabel('Speed Ratio');
1424
1425    %=================================================================================
    ==
1426    %=================================================================================
    ==
1427
1428    elseif strcmp(action,'cwrite')
1429       Hndllist = get(gca,'Userdata');
1430       closeHndl = Hndllist(9);
1431       pradHndl = Hndllist(12);
1432
1433       prad=get(pradHndl,'Userdata');
1434
1435       trkslp = get(closeHndl,'Userdata');
1436       trkslpl = trkslp(:,1);
1437       trkslpr = trkslp(:,2);
1438
1439       figure('Name','Trackslips',...
1440            'NumberTitle','off',...
1441            'BackingStore','off',...
1442            'Visible','on',...
1443            'Units','normalized',...
1444            'Position',[0.05 0.05 0.9 0.9]);
1445        plot(t,trkslpl,t,trkslpr);
1446        legend('Left Track','Right Track');
1447        title(['Track Speed Ratios (' num2str(prad) ' in. Pivot Radius)']);
1448        xlabel('Time (s)');
1449        ylabel('Speed Ratio');
1450    end;
```

tscalc.m

```
1   %  TSCALC - Track Slip Calculator
2   %    Plots the vehicle paths as calculated by YAW/SPEED
3   %  and L2TRK/R2TRK.  Allows user to change the time and
4   %  adjust the track slip of each track individually.  Will
5   %  then replot the data.  Allows writing and loading of
6   %  track slip data files.
7   %
8   %  Ex.
9   %  tscalc(t,speed,yaw,l2trk,r2trk);
10
11      function tscalc(t,speed,yaw,l2trk,r2trk,action)
12      if nargin==5
13
14          L=max(size(t));
15          % for m=1:floor(L/100)-1
16          %     t_(m)=mean(t(100*(m-1)+1:100*m));
17          %     speed_(m)=mean(speed(100*(m-1)+1:100*m));
18          %     yaw_(m)=mean(yaw(100*(m-1)+1:100*m));
19          %     l2trk_(m)=mean(l2trk(100*(m-1)+1:100*m));
20          %     r2trk_(m)=mean(r2trk(100*(m-1)+1:100*m));
```

```
21          % end;
22
23          pradNumber=figure(...
24              'Name','Pivot Radius',...
25              'NumberTitle','off',...
26              'BackingStore','off',...
27              'Visible','on',...
28              'Units','normalized',...
29              'Position',[0.44 0.45 0.12 0.1],...
30              'MenuBar','none');
31
32
33          %==================================
34          % The Pivot Radius button
35          btnNumber=7;
36          labelStr='50';
37          cmdStr='Close';
38          callbackstr='tscalc(t,speed,yaw,l2trk,r2trk,''prad'');';
39
40          %Generic button information
41          btnPos=[.25 .25 .5 .25];
42          pradHndl=uicontrol(...
43              'Style','edit',...
44              'Units','normalized',...
45              'Position',btnPos,...
46              'String',labelStr,...
47              'Interruptible','on',...
48              'Enable','on',...
49              'Userdata',50,...
50              'Callback',callbackstr);
51
52          %==================================
53          % Text
54          btnPos=[.01 .55 .98 .3];
55          uicontrol(...
56              'Style','text',...
57              'Units','normalized',...
58              'Position',btnPos,...
59              'String',['Enter Pivot Radius in ''inches''.'],...
60              'BackgroundColor',[.8 .8 .8]);
61
62          set(gcf,'Userdata',pradHndl)
63
64      elseif strcmp(action,'calc')
65          pradNumber=get(gcf,'Userdata');
66          prad=get(pradNumber,'Userdata');
67
68          close(gcf);
69
70          speed=speed.*88./60;
71
72          yaw=pi.*yaw./180;
73          speed=(63./12)*yaw+speed;
74
75          sl=speed+prad.*yaw./12;
76          sr=speed-prad.*yaw./12;
77          a=find(sl<0.005&sl>-0.005);
78          b=find(sr<0.005 & sr>-0.005);
79
80          sl=sl.*60./88;
81          sr=sr.*60./88;
82
83          tsl=l2trk./sl;
84          tsr=r2trk./sr;
85
86          c=find(tsl<0.005&tsl>-0.005);
87          d=find(tsr<0.005&tsr>-0.005);
88
89          tsl(a)=1e+0;
90          tsr(b)=1e+0;
91          tsl(c)=1e-10;
```

```
92          tsr(d)=1e-10;
93
94          pradvect = zeros(size(tsl));
95          pradvect(1) = prad;
96
97          % tsl=spline(t_,tsl,t);
98          % tsr=spline(t_,tsr,t);
99
100         [filename,pathname] = uiputfile('*.ASC','Select ASCII file to SAVE');
101
102         lerrNumber=figure(...
103                     'Name','Saving...',...
104                     'NumberTitle','off',...
105                     'BackingStore','off',...
106                     'Visible','on',...
107                     'Units','normalized',...
108                     'Position',[0.44 0.45 0.12 0.1],...
109                     'MenuBar','none');
110
111                 %==================================
112                 % Text
113                 btnPos=[.01 .55 .98 .3];
114                 uicontrol(...
115                     'Style','text',...
116                     'Units','normalized',...
117                     'Position',btnPos,...
118                     'String',['Saving to disk.  Please Wait'],...
119                     'BackgroundColor',[.8 .8 .8]);
120
121         if filename~=0
122             dlmwrite([pathname filename],[t tsl tsr pradvect],'\t');
123         end;
124
125         close(gcf);
126
127     elseif strcmp(action,'prad')
128
129         pradNumber=get(gcf,'Userdata');
130         a=get(pradNumber,'Userdata');
131         b=get(pradNumber,'String');
132         c=str2double(b);
133
134         if (isequalwithequalnans(NaN,c) | c<=0)
135
136             lerrNumber=figure(...
137                 'Name','Pivot Radius Error',...
138                 'NumberTitle','off',...
139                 'BackingStore','off',...
140                 'Visible','on',...
141                 'Units','normalized',...
142                 'Position',[0.44 0.45 0.12 0.1],...
143                 'MenuBar','none');
144
145             %==================================
146             % The OK button
147             btnNumber=7;
148             labelStr='Ok';
149             cmdStr='Close';
150             callbackstr='close(gcf);';
151
152             %Generic button information
153             btnPos=[.25 .25 .5 .25];
154             setHndl=uicontrol(...
155                 'Style','pushbutton',...
156                 'Units','normalized',...
157                 'Position',btnPos,...
158                 'String',labelStr,...
159                 'Interruptible','on',...
160                 'Enable','on',...
161                 'Callback',callbackstr);
162
```

```
163             %==================================
164             % Text
165             btnPos=[.01 .55 .98 .3];
166             uicontrol(...
167                 'Style','text',...
168                 'Units','normalized',...
169                 'Position',btnPos,...
170                 'String',['Given radius is not a positive real number.  '...
171                     'Do not use units.  Please retry.'],...
172                 'BackgroundColor',[.8 .8 .8]);
173 
174             set(pradNumber,'String',num2str(a));
175         else
176             set(pradNumber,'Userdata',c);
177             tscalc(t,speed,yaw,l2trk,r2trk,'calc');
178         end;
179     else
180         pradNumber=figure(...
181             'Name','Pivot Radius',...
182             'NumberTitle','off',...
183             'BackingStore','off',...
184             'Visible','off',...
185             'Units','normalized',...
186             'Position',[0.44 0.45 0.12 0.1],...
187             'MenuBar','none');
188 
189 
190         %==================================
191         % The Pivot Radius button
192         btnNumber=7;
193         labelStr='50';
194         cmdStr='Close';
195         callbackstr='tscalc(t,speed,yaw,l2trk,r2trk,''prad'');';
196 
197         %Generic button information
198         btnPos=[.25 .25 .5 .25];
199         pradHndl=uicontrol(...
200             'Style','edit',...
201             'Units','normalized',...
202             'Position',btnPos,...
203             'String',labelStr,...
204             'Interruptible','on',...
205             'Enable','on',...
206             'Userdata',50,...
207             'Callback',callbackstr);
208 
209 
210         set(pradNumber,'Userdata',pradHndl);
211         set(pradHndl,'Userdata',action);
212         tscalc(t,speed,yaw,l2trk,r2trk,'calc');
213     end;
```

crvwrite.m

```
1   %  CRVWRITE - Curve Writer
2   %    Writes selected portions of the curve as a separate file
3   %  with time, X-Y, and track slip data.  Also writes a separate
4   %  info file that describes the data file.  CRVWRITE is only to
5   %  be used by TRKSLPCALC.
6 
7   function crvwrite(action)
8   if nargin==0
9        oldFig=gcf;
10          Hndllist = get(gca,'Userdata');
11          figNumber = Hndllist(1);
12          axHndl = Hndllist(2);
13          textHndl = Hndllist(3);
```

```
14          time1Hndl = Hndllist(4);
15          time2Hndl = Hndllist(5);
16          time1edHndl = Hndllist(6);
17          time2edHndl = Hndllist(7);
18          plotHndl = Hndllist(8);
19          closeHndl = Hndllist(9);
20          loadHndl = Hndllist(10);
21          writeHndl = Hndllist(11);
22          pradHndl = Hndllist(12);
23          tscalcHndl = Hndllist(13);
24          tsplotHndl = Hndllist(14);
25          cwriteHndl = Hndllist(15);
26          time3Hndl = Hndllist(16);
27          time3edHndl = Hndllist(17);
28
29          mainmat = get(cwriteHndl,'Userdata');
30
31          t1 = get(time1Hndl,'Value');
32          t2 = get(time2Hndl,'Value');
33          t3 = get(time3Hndl,'Value');
34
35          t=mainmat(:,1);
36
37          indx1=round(mean(find(mainmat(:,1)==get(time1Hndl,'Value'))));
38          indx2=round(mean(find(mainmat(:,1)==get(time2Hndl,'Value'))));
39          indx3=round(mean(find(mainmat(:,1)==get(time3Hndl,'Value'))));
40
41
42          set(oldFig,'Visible','off');
43
44          newFig=figure(...
45              'Name','Curve Writer Information Selection',...
46              'NumberTitle','off',...
47              'MenuBar','none',...
48              'BackingStore','off',...
49              'Visible','off',...
50              'Units','normalized',...
51              'Position',[0.2 0.4 0.6 0.2],...
52              'CloseRequestFcn','');
53          colordef(newFig,'black');
54
55          %===================================
56          % Information for buttons
57          labelColor=[0.8 0.8 0.8];
58          yInitPos=0.50;
59          xPos=0.05;
60          btnLen=0.10;
61          btnWid=0.10;
62          txtWid=0.065;
63          editWid=0.09;
64          sliderLen=0.2;
65          txtLen=0.05;
66          spacing=0.05;
67
68
69
70          %===================================
71          % The CONSOLE frame #1
72          frmBorder=0.02;
73          yPos=.285;
74          frmPos=[xPos-.025 yPos .95 .665];
75          h=uicontrol('Style','frame',...
76              'Units','normalized',...
77              'Position',frmPos,...
78              'BackgroundColor',[0.5 0.5 0.5]);
79
80          %===================================
81          % The Close button
82          btnNumber=1;
83          yPos=0.1;
84          labelStr='Close';
```

```
85          cmdStr='close';
86          callbackstr='crvwrite(''close'');';
87
88          %Generic button information
89          btnPos=[.9-.5*btnLen yPos btnLen btnWid];
90          closeHndl=uicontrol(...
91              'Style','pushbutton',...
92              'Units','normalized',...
93              'Position',btnPos,...
94              'String',labelStr,...
95              'Interruptible','on',...
96              'Callback',callbackstr);
97
98          %==================================
99          % The Write button
100         btnNumber=2;
101         yPos=0.1;
102         labelStr='Write';
103         cmdStr='write';
104         callbackstr='crvwrite(''write'');';
105
106         %Generic button information
107         btnPos=[.9-1.5*btnLen-spacing yPos btnLen btnWid];
108         writeHndl=uicontrol(...
109             'Style','pushbutton',...
110             'Units','normalized',...
111             'Position',btnPos,...
112             'String',labelStr,...
113             'Interruptible','on',...
114             'Enable','off',...
115             'Callback',callbackstr);
116
117         %==================================
118         % The Title edit
119         btnNumber=3;
120         yPos=0.1;
121         labelStr='*Replace with acceptable title*';
122         cmdStr='title';
123
124         %Generic button information
125         btnPos=[xPos .735 .9 editWid];
126         titleHndl=uicontrol(...
127             'Style','edit',...
128             'Units','normalized',...
129             'Position',btnPos,...
130             'String',labelStr,...
131             'HorizontalAlignment','left',...
132             'Interruptible','on');
133
134         %==================================
135         % The Description edit
136         btnNumber=4;
137         yPos=0.1;
138         labelStr='*Replace with description of enclosed data*';
139         cmdStr='desc';
140
141         %Generic button information
142         btnPos=[xPos .535 .9 editWid];
143         descHndl=uicontrol(...
144             'Style','edit',...
145             'Units','normalized',...
146             'Position',btnPos,...
147             'String',labelStr,...
148             'HorizontalAlignment','left',...
149             'Interruptible','on');
150
151         %==================================
152         % The Curve Type Popup
153         btnNumber=5;
154         yPos=0.1;
155         labelStr='Straightaway|Left Curve|Right Curve|S-Curve(L-R)|S-Curve(R-L)';
```

```
156         cmdStr='desc';
157 
158         %Generic button information
159         btnPos=[xPos .335 .3 editWid];
160         ctypeHndl=uicontrol(...
161             'Style','popup',...
162             'Units','normalized',...
163             'Position',btnPos,...
164             'String',labelStr,...
165             'Interruptible','on');
166 
167         %==================================
168         % The Ready Checkbox
169         btnNumber=5;
170         yPos=0.08;
171         cmdStr='desc';
172         callbackstr='crvwrite(''ready'');';
173 
174         %Generic button information
175         btnPos=[.625-.07*btnLen yPos .14*btnLen .6*btnWid];
176         readyHndl=uicontrol(...
177             'Style','checkbox',...
178             'Units','normalized',...
179             'Position',btnPos,...
180             'Max',1,'Min',0,...
181             'Interruptible','on',...
182             'Callback',callbackstr);
183 
184         %==================================
185         % Text
186         btnPos=[xPos 0.835 2.*btnLen txtWid];
187         uicontrol(...
188             'Style','text',...
189             'Units','normalized',...
190             'Position',btnPos,...
191             'HorizontalAlignment','left',...
192             'String','Title (First line of information file):',...
193             'BackgroundColor',[.5 .5 .5]);
194 
195         btnPos=[xPos 0.635 2.*btnLen txtWid];
196         uicontrol(...
197             'Style','text',...
198             'Units','normalized',...
199             'Position',btnPos,...
200             'HorizontalAlignment','left',...
201             'String','Description:',...
202             'BackgroundColor',[.5 .5 .5]);
203 
204         btnPos=[xPos 0.435 2.*btnLen txtWid];
205         uicontrol(...
206             'Style','text',...
207             'Units','normalized',...
208             'Position',btnPos,...
209             'HorizontalAlignment','left',...
210             'String','Curve Type:',...
211             'BackgroundColor',[.5 .5 .5]);
212 
213         btnPos=[.6 .15 txtLen txtWid];
214         uicontrol(...
215             'Style','text',...
216             'Units','normalized',...
217             'Position',btnPos,...
218             'String','Ready?',...
219             'BackgroundColor',[.5 .5 .5]);
220 
221         set(newFig,'Visible','on','Userdata',[closeHndl,writeHndl,...
222                 titleHndl descHndl ctypeHndl readyHndl]);
223         set(closeHndl,'Userdata',[oldFig, newFig]);
224         set(writeHndl,'Userdata',mainmat);
225         set(ctypeHndl,'Userdata',[t1 t2 t3 indx1 indx2 indx3]);
226 
```

```
    %================================================================================
==
    %================================================================================
==

    elseif strcmp(action,'close')
        Hndllist = get(gcf,'Userdata');
        closeHndl = Hndllist(1);
        figHndl = get(closeHndl,'Userdata');

        oldFig = figHndl(1);
        newFig = figHndl(2);

        delete(newFig);
        set(oldFig,'Visible','on');

    %================================================================================
==
    %================================================================================
==

    elseif strcmp(action,'ready')
        Hndllist = get(gcf,'Userdata');
        writeHndl = Hndllist(2);
        readyHndl = Hndllist(6);

        if get(readyHndl,'Value')==0
            set(writeHndl,'Enable','off');
        elseif get(readyHndl,'Value')==1
            set(writeHndl,'Enable','on');
        end

    %================================================================================
==
    %================================================================================
==

    elseif strcmp(action,'write')
        Hndllist = get(gcf,'Userdata');
        closeHndl = Hndllist(1);
        writeHndl = Hndllist(2);
        titleHndl = Hndllist(3);
        descHndl = Hndllist(4);
        ctypeHndl = Hndllist(5);
        readyHndl = Hndllist(6);

        figHndl = get(closeHndl,'Userdata');
        oldFig = figHndl(1);
        newFig = figHndl(2);

        mainmat = get(writeHndl,'Userdata');

        tindx = get(ctypeHndl,'Userdata');
        t1=tindx(1);
        t2=tindx(2);
        t3=tindx(3);
        indx1=tindx(4);
        indx2=tindx(5);
        indx3=tindx(6);

        t=mainmat(:,1);
        cgx2=mainmat(:,2);
        cgy2=mainmat(:,3);
        trkslpl=mainmat(:,4);
        trkslpr=mainmat(:,5);
        speed=mainmat(:,6);
        yaw=mainmat(:,7);

        [filename,pathname] = uiputfile('*.ASC','Select ASCII file to SAVE');

        if filename==0
```

```
292             crvwrite('close');
293             return;
294         end
295         lerrNumber=figure(...
296             'Name','Saving...',...
297             'NumberTitle','off',...
298             'BackingStore','off',...
299             'Visible','on',...
300             'Units','normalized',...
301             'Position',[0.425 0.48 0.15 0.04],...
302             'MenuBar','none',...
303             'CloseRequestFcn','');
304 
305         %================================
306         % Text
307         btnPos=[.01 .55 .98 .3];
308         uicontrol(...
309             'Style','text',...
310             'Units','normalized',...
311             'Position',btnPos,...
312             'String',['Saving to disk.  Please Wait...'],...
313             'BackgroundColor',[.8 .8 .8]);
314 
315         fid = fopen([pathname 'info_' filename],'w');
316         fprintf(fid,['Information File Relating to ''' filename '''.\n\n']);
317         fprintf(fid,['Title:\t' get(titleHndl,'String') '\n\n']);
318         fprintf(fid,['Description:\t' get(descHndl,'String') '\n\n']);
319         fprintf(fid,['Start Time:\t' num2str(t1) '\ts\n']);
320         fprintf(fid,['End Time:\t' num2str(t2) '\ts\n\n']);
321         fprintf(fid,'Curve Specific Data:\n');
322         fprintf(fid,'********************\n');
323 
324 
325 
326         switch get(ctypeHndl,'Value')
327             case 1
328                 fprintf(fid,'Curve Type:\tStraightaway\n');
329                 a=cumtrapz(t(indx1:indx2),speed(indx1:indx2));
330                 fprintf(fid,['Length of Straightaway:\t' num2str(max(a)) '\tft\n']);
331                 fprintf(fid,['Average Speed:\t' num2str(mean(speed(indx1:indx2))) ...
332                         '\tft/s\n\t' num2str(60*mean(speed(indx1:indx2))/88)
    '\tmph\n']);
333             case 2
334                 fprintf(fid,'Curve Type:\tLeft Curve\n');
335                 r=-mean(speed(indx1:indx2))/mean(yaw(indx1:indx2));
336                 fprintf(fid,['Radius of Curvature:\t' num2str(r) '\tft\n']);
337                 theta=max(-cumtrapz(t(indx1:indx2),yaw(indx1:indx2)));
338                 fprintf(fid,['Total Angle Covered:\t' num2str(theta) '\tdeg\n']);
339                 fprintf(fid,['Average Speed:\t' num2str(mean(speed(indx1:indx2))) ...
340                         '\tft/s\n\t' num2str(60*mean(speed(indx1:indx2))/88)
    '\tmph\n']);
341             case 3
342 
343 
344 
345         end
346 
347 
348 
349 
350 
351     %
352     %
353     % close(gcf);
354     %
355     % speed=speed.*88./60;
356     %
357     % yaw=pi.*yaw./180;
358     %
359     % sl=speed+prad.*yaw./12;
360     % sr=speed-prad.*yaw./12;
```

```
% a=find(sl<0.005&sl>-0.005);
% b=find(sr<0.005 & sr>-0.005);
%
% sl=sl.*60./88;
% sr=sr.*60./88;
%
% tsl=l2trk./sl;
% tsr=r2trk./sr;
%
% c=find(tsl<0.005&tsl>-0.005);
% d=find(tsr<0.005&tsr>-0.005);
%
% tsl(a)=1e+0;
% tsr(b)=1e+0;
% tsl(c)=1e-10;
% tsr(d)=1e-10;
%
% pradvect = zeros(size(tsl));
% pradvect(1) = prad;
%
% % tsl=spline(t_,tsl,t);
% % tsr=spline(t_,tsr,t);
%
% [filename,pathname] = uiputfile('*.ASC','Select ASCII file to SAVE');
%
% lerrNumber=figure(...
%     'Name','Saving...',...
%     'NumberTitle','off',...
%     'BackingStore','off',...
%     'Visible','on',...
%     'Units','normalized',...
%     'Position',[0.44 0.45 0.12 0.1],...
%     'MenuBar','none');
%
% %==================================
% % Text
% btnPos=[.01 .55 .98 .3];
% uicontrol(...
%     'Style','text',...
%     'Units','normalized',...
%     'Position',btnPos,...
%     'String',['Saving to disk.  Please Wait'],...
%     'BackgroundColor',[.8 .8 .8]);
%
% if filename~=0
%     dlmwrite([pathname filename],[t tsl tsr pradvect],'\t');
% end;
%
% close(gcf);
%
% elseif strcmp(action,'prad')
%
%     pradNumber=get(gcf,'Userdata');
%     a=get(pradNumber,'Userdata');
%     b=get(pradNumber,'String');
%     c=str2double(b);
%
%     if (isequalwithequalnans(NaN,c) | c<=0)
%
%         lerrNumber=figure(...
%             'Name','Pivot Radius Error',...
%             'NumberTitle','off',...
%             'BackingStore','off',...
%             'Visible','on',...
%             'Units','normalized',...
%             'Position',[0.44 0.45 0.12 0.1],...
%             'MenuBar','none');
%
%         %==================================
%         % The OK button
%         btnNumber=7;
```

```
%         labelStr='Ok';
%         cmdStr='Close';
%         callbackstr='close(gcf);';
%
%         %Generic button information
%         btnPos=[.25 .25 .5 .25];
%         setHndl=uicontrol(...
%             'Style','pushbutton',...
%             'Units','normalized',...
%             'Position',btnPos,...
%             'String',labelStr,...
%             'Interruptible','on',...
%             'Enable','on',...
%             'Callback',callbackstr);
%
%         %==================================
%         % Text
%         btnPos=[.01 .55 .98 .3];
%         uicontrol(...
%             'Style','text',...
%             'Units','normalized',...
%             'Position',btnPos,...
%             'String',['Given radius is not a positive real number.  '...
%                 'Do not use units.  Please retry.'],...
%             'BackgroundColor',[.8 .8 .8]);
%
%         set(pradNumber,'String',num2str(a));
%     else
%         set(pradNumber,'Userdata',c);
%         tscalc(t,speed,yaw,l2trk,r2trk,'calc');
    status = fclose(fid);
    delete(lerrNumber);
    end;
```

pscalc1.m

```
% PSCALC1 - Perfect Simulation Calculator
%   Simulates an oval track with 90 ft. straightaways and
% 60 ft. radius turns using algorithms.


function pscalc
t=(0:0.01:39.13);
dt=0.01;
v1=15*ones(size(t));
    w1=zeros(size(t));
    w1(201:1457)=15/60*ones(size(w1(201:1457)));
    w1(2058:3314)=15/60*ones(size(w1(2058:3314)));
    v2(1)=15;
    w2(1)=0;
    gam(1)=0;
    a=pwd;
    udatmat=csvread([a '\pathclear.csv']);
    l1=udatmat(2)+udatmat(3);
    l2=udatmat(4);

    for n=2:length(t)
        v2(n)=cos(gam(n-1))*v1(n)+sin(gam(n-1))*l1*w1(n);
        w2(n)=(sin(gam(n-1))*v1(n)-cos(gam(n-1))*l1*w1(n))/l2;
        gam(n)=gam(n-1)+dt*(w1(n)-w2(n));
    end;

    % subplot(2,1,1);
    % plot(t,v1,t,v2);
    % subplot(2,1,2);
    % plot(t,w1,t,w2);

```

```
    dist1=dt*cumtrapz(v1);
    ddist1=[dist1(1),diff(dist1)];
    dist2=dt*cumtrapz(v2);
    ddist2=[dist2(1),diff(dist2)];

    th1=dt*cumtrapz(w1);
    th2=dt*cumtrapz(w2);

    dx1=ddist1.*sin(th1);
    dy1=ddist1.*cos(th1);
    dx2=ddist2.*sin(th2);
    dy2=ddist2.*cos(th2);

    x1=cumsum(dx1);
    y1=cumsum(dy1);
    x2=cumsum(dx2);
    y2=cumsum(dy2)-l1-l2;

    figure('Name','Vehicle Path (Oval Track)','NumberTitle','off',...
        'Position',[0 0 1024 768]);
    movegui('center');
    lv=line(x1,y1,'Color',[0 0 .5]);
    tv=line(x2,y2,'Color',[0 .5 0]);
    strtpt=line(x1(1),y1(1),'Color',[.5 0 0],...
        'LineStyle','none','Marker','o');
    endpt=line(x1(3914),y1(3914),'Color',[.5 0 0],...
        'LineStyle','none','Marker','x','MarkerSize',12);
    axis equal;
    legend('Lead Vehicle CM','Trail Vehicle CM','Start','Finish');
    title('Vehicle Path (Oval Track)');
    xlabel('(ft.)');
    ylabel('(ft.)');

    figure('Name','Angles (Oval Track)','NumberTitle','off',...
        'Position',[0 0 1024 768]);
    movegui('center');
    subplot(2,1,1);
    plot(t,180/pi*th1,t,180/pi*th2);
    title('Vehicle Angles');
    xlabel('Time (s.)');
    ylabel('Angle (deg.)');
    legend('Lead Vehicle','Trail Vehicle');
    subplot(2,1,2);
    plot(t,180*gam/pi);
    title('Intervehicular Angle (\gamma)');
    xlabel('Time (s.)');
    ylabel('Angle (deg.)');
```

corrcalc.m

```
% CORRCALC - Correlation Program
%    Correlates the instantaneous center theory with experimental data
% collected during the teleop testing on July 15th and 16th, 2003.
% Use the MODCAT program to load experimental data. Type HELP MODCAT
% for assistance.
%
% Ex.
%
[v1,v2_1,v2,gam,gam2,omega1,omega2,gam_dot,gam_dot2,x1,y1,x2,y2]=corrcalc(t,dt,speed,yaw,ldist,loaded_file,'yes');

    function [v1,v2_1,v2,gam,gam2,omega1,omega2,gam_dot,gam_dot2,x1,y1,x2,y2]=corrcalc(t,dt,speed,yaw,ldist,loaded_file,prntopt)

    if nargin==6
        prntopt='nosave/nocut';
```

```
    end;

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %  Experimental Calculations
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    a=pwd;
    a=a(1:max(find(a=='\')));
    udatmat=csvread([a 'pathclear.csv']);
    l2=udatmat(2);
    l3=udatmat(3);
    l4=udatmat(4);
    yaw=yaw*pi/180;
    speed=88/60*speed;
    speed=63/12*yaw+speed;
    v2_1=speed;
    dist=dt*cumsum(speed);
    th2=dt*cumsum(yaw);


    ddist=[dist(1);diff(dist)];
    dx=ddist.*sin(th2);
    dy=ddist.*cos(th2);
    x2=cumsum(dx);
    y2=cumsum(dy);


    l_1=sqrt(4.10^2+2.05^2);
    alph=acos((l_1^2+1.25^2-(sqrt(4.1^2+0.80^2)+(ldist-32.7)/12).^2)/(2.5*l_1));
    alph_o=atan(2);
    gam=alph-alph_o;

    th1=th2+gam;

    x1=x2+l4*sin(th2)+(l2+l3)*sin(th1);
    y1=y2+l4*cos(th2)+(l2+l3)*cos(th1);


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %  Theoretical Calculations
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    a=diff(gam)/dt;
    gam_dot=[a(1);a];
    a=diff(x1);
    b=diff(y1);
    c=(a.^2+b.^2).^.5;

    v1=[c(1);c];
    v1=v1./dt;
    omega1=yaw+gam_dot;

    % omega2=(v1.*sin(gam)-omega1.*(l2+l3).*cos(gam))/l4;
    %
    % v2=v1.*cos(gam)+omega1.*(l2+l3).*sin(gam);
    %
    % gam_dot2=omega1-omega2;
    po=length(prntopt);
    tl=length(t);

    if strcmp(prntopt(po-3:po),'/cut')
        t=t(100:tl);
        x1=x1(100:tl);
        x2=x2(100:tl);
        y1=y1(100:tl);
        y2=y2(100:tl);
        v2_1=v2_1(100:tl);
        yaw=yaw(100:tl);
        omega1=omega1(100:tl);
        gam=gam(100:tl);
        th1=th1(100:tl);
    end;
```

```
85
86
87      v2(1)=v2_1(1);
88      omega2(1,1)=yaw(1);
89      gam2(1)=gam(1);
90      gam_dot2(1)=omega1(1)-omega2(1);
91
92      figure('NumberTitle','off','Name','Progress...','MenuBar','none',...
93          'Position',[0 0 200 100]);
94      movegui(gcf,'center');
95      h=uicontrol('Style','Text','Position',[25 40 150 20],...
96          'HorizontalAlignment','center','String','0%',...
97          'BackgroundColor',get(gcf,'Color'));
98
99      for n=2:length(t)
100         gam2(n,1)=gam2(n-1)+dt*gam_dot2(n-1);
101         v2(n,1)=v1(n).*cos(gam2(n))+omega1(n).*(l2+l3).*sin(gam2(n));
102         omega2(n,1)=(v1(n).*sin(gam2(n))-omega1(n)*(l2+l3).*cos(gam2(n)))/l4;
103         gam_dot2(n,1)=omega1(n)-omega2(n);
104         set(h,'String',[num2str(n/length(t)*100,'%3.0f') '%']);
105         set(gcf,'Name',['Progress... ' num2str(n/length(t)*100,'%3.0f') '%']);
106         drawnow;
107     end;
108     close(gcf);
109
110     th2_2=th1-gam2;
111     x2_2=x1-l4*sin(th2_2)-(l2+l3)*sin(th1);
112     y2_2=y1-l4*cos(th2_2)-(l2+l3)*cos(th1);
113
114     figure('Name','Close All','NumberTitle','off','Menubar','none',...
115         'Position',[50 50 200 100]);
116     % movegui(gcf,'center');
117     uicontrol('Style','Pushbutton','Position',[25 25 150 50],...
118         'String','Close All','Callback','close all');
119
120     % figure('Name','Second Vehicle Speeds','NumberTitle','off',...
121     %     'Menubar','none','Position',[0 0 1024 768]);
122     % plot(t,speed,t,v2);
123     % legend('Measured Speed','Theoretical Speed');
124     % movegui(gcf,'center');
125     %
126     % figure('Name','Second Vehicle Speed Difference','NumberTitle','off',...
127     %     'Menubar','none','Position',[0 0 1024 768]);
128     % plot(t,speed-v2);
129     % movegui(gcf,'center');
130     %
131     % figure('Name','Second Vehicle Yaw Rates','NumberTitle','off',...
132     %     'Menubar','none','Position',[0 0 1024 768]);
133     % plot(t,yaw,t,omega2);
134     % legend('Measured Yaw','Theoretical Yaw');
135     % movegui(gcf,'center');
136     %
137     % figure('Name','Second Vehicle Yaw Difference','NumberTitle','off',...
138     %     'Menubar','none','Position',[0 0 1024 768]);
139     % plot(t,yaw-omega2);
140     % movegui(gcf,'center');
141     %
142
143     a=figure('Name','Vehicle Path','NumberTitle','off',...
144         'Position',[0 0 1024 768]);
145     plot(x1,y1,x2,y2,x2_2,y2_2);
146     axis equal;
147     movegui(gcf,'center');
148     legend('L.V.','Exp. T.V.','Opt. T.V.');
149     title('Vehicle Path');
150     xlabel('(ft.)');
151     ylabel('(ft.)');
152
153
154
155     if strcmp(prntopt(1:4),'save')
```

```
156         b=figure('Name','Saving...','NumberTitle','off',...
157             'Position',[0 0 200 100],'MenuBar','none');
158         uicontrol('Style','Text','Position',[25 40 150 20],...
159             'HorizontalAlignment','center',...
160             'String',['Saving ' loaded_file '_path.jpg'],...
161             'BackgroundColor',get(gcf,'Color'));
162         movegui(gcf,'center');
163         print(['-f' num2str(a)],'-djpeg','-r72',[loaded_file '_path.jpg']);
164         close(b);
165     end;
166 
167 
168     %
169     a=figure('Name','gam_dot2','NumberTitle','off',...
170         'Position',[0 0 1024 768]);
171     plot(t,180/pi*gam,t,180/pi*gam2);
172     movegui(gcf,'center');
173     legend('Experimental','Optimal');
174     title('Gamma comparison');
175     xlabel('Time (sec.)');
176     ylabel('Angle (deg.)');
177 
178 
179     if strcmp(prntopt(1:4),'save')
180         b=figure('Name','Saving...','NumberTitle','off',...
181             'Position',[0 0 200 100],'MenuBar','none');
182         uicontrol('Style','Text','Position',[25 40 150 20],...
183             'HorizontalAlignment','center',...
184             'String',['Saving ' loaded_file '_gamma.jpg'],...
185             'BackgroundColor',get(gcf,'Color'));
186         movegui(gcf,'center');
187         print(['-f' num2str(a)],'-djpeg','-r72',[loaded_file '_gamma.jpg']);
188         close(b);
189     end;
```

ts_least_sq2.m

```
1   %TS_LEAST_SQ2    Least Squares Calculator
2   %   Calculates the least squares solution for
3   %   trackslip as a function of velocity and radius
4   %   of curvature implementing the logarithmic
5   %   value calculated with the function "ts_least_sq_calc".
6   %   To see the trackslip approximation function,
7   %   type: "ts_least_sq2('equation')" at the command
8   %   prompt.
9 
10      function ts_least_sq2(fname);
11      if nargin==0
12          close all;
13          ts_least_sq2('lt_filelist.txt');
14          ts_least_sq2('rt_filelist.txt');
15      elseif strcmp('equation',fname);
16          figure('Units','normalized',...
17              'Position',[.25 .4 .5 .2],...
18              'MenuBar','none',...
19              'NumberTitle','off',...
20              'Name','Trackslip Approximation Function');
21          axes('Units','normalized',...
22              'Position',[0 0 1 1],'Visible','off');
23          text('FontSize',14,...
24              'units','normalized',...
25              'Position',[.5 .9],...
26              'HorizontalAlignment','center',...
27              'String','For Inside Left Turn: \bff(\nu,(\itr))=a_(1)\nuln((\itr)-
    22.4170)+a_(2)ln((\itr)-22.4170)+a_(3)\nu+a_(4)\approx(\itTS_(left))');
28          text('FontSize',14,...
29              'units','normalized',...
30              'Position',[.5 .75],...
31              'HorizontalAlignment','center',...
```

```
            'String','For Inside Right Turn: \bff(\nu,(\itr))=a_(1)\nuln((\itr)-
25.7639)+a_(2)ln((\itr)-25.7639)+a_(3)\nu+a_(4)\approx(\itTS_(right))');
        text('FontSize',14,...
            'units','normalized',...
            'Position',[.5 .6],...
            'HorizontalAlignment','center',...
            'String','For Outside Turns: \bff(\nu,(\itr))=a_(1)\nuln((\itr)-
25.7639)+a_(2)ln((\itr)-25.7639)+a_(3)\nu+a_(4)\approx(\itTS_(right))');
        text('units','normalized',...
            'Position',[.35 .4],...
            'HorizontalAlignment','left',...
            'String','where:');
        text('units','normalized',...
            'Position',[.4 .32],...
            'HorizontalAlignment','left',...
            'String','\epsilon');
        text('units','normalized',...
            'Position',[.43 .32],...
            'HorizontalAlignment','left',...
            'String','=');
        text('units','normalized',...
            'Position',[.45 .32],...
            'HorizontalAlignment','left',...
            'String','error');
        text('units','normalized',...
            'Position',[.4 .24],...
            'HorizontalAlignment','left',...
            'String','\nu');
        text('units','normalized',...
            'Position',[.43 .24],...
            'HorizontalAlignment','left',...
            'String','=');
        text('units','normalized',...
            'Position',[.45 .24],...
            'HorizontalAlignment','left',...
            'String','velocity');
        text('units','normalized',...
            'Position',[.4 .16],...
            'HorizontalAlignment','left',...
            'String','(\itr)');
        text('units','normalized',...
            'Position',[.43 .16],...
            'HorizontalAlignment','left',...
            'String','=');
        text('units','normalized',...
            'Position',[.45 .16],...
            'HorizontalAlignment','left',...
            'String','radius of curvature');
    else
        filelist=textread(fname,...
            '%s','delimiter','\t','whitespace','');
        filelist=char(filelist);

        if strcmp(fname,'lt_filelist.txt')
            turndir='Left Turn';
            intrkside='Left';
            outtrkside='Right';
            m=22.4170;
        else
            turndir='Right Turn';
            intrkside='Right';
            outtrkside='Left';
            m=25.7639;
        end;

        main=zeros(round(max(size(filelist))/2),4);

        for n=1:round(max(size(filelist))/2);
            test=dlmread([pwd filelist(2*n-1,:) filelist(2*n,:)],'\t');
            main(n,1)=mean(test(:,4));
            main(n,2)=mean(test(:,5));
```

```
            test=textread([pwd filelist(2*n-1,:) 'info_' filelist(2*n,:)],...
                '%s','delimiter','\t','whitespace','');
            test=char(test);
            chk=regexp(test,'Radius of Curvature');
            chk=cellfun('length',chk);
            a=find(chk>0);
            main(n,3)=str2double(test(a+1,:));
            chk=regexp(test,'Average Speed');
            chk=cellfun('length',chk);
            a=find(chk>0);
            main(n,4)=str2double(test(a+1,:));
        end;

        v=main(:,4);
        r=main(:,3);
        if strcmp(turndir,'Left Turn');
            tsi=main(:,1);
            tso=main(:,2);
        else
            tsi=main(:,2);
            tso=main(:,1);
        end;

        length=max(size(main(:,2)));

        v2l2=v.^2.*log(r-m).^2;
        vl2=v.*log(r-m).^2;
        v2l=v.^2.*log(r-m);
        vl=v.*log(r-m);
        l2=log(r-m).^2;
        l=log(r-m);
        v2=v.^2;
        vltsi=v.*log(r-m).*tsi;
        ltsi=log(r-m).*tsi;
        vtsi=v.*tsi;
        v2r2=v.^2.*r.^2;
        v2r=v.^2.*r;
        vr2=v.*r.^2;
        vr=v.*r;
        r2=r.^2;
        vrtso=v.*r.*tso;
        vtso=v.*tso;
        rtso=r.*tso;

        a=inv([sum(v2l2),sum(vl2),sum(v2l),sum(vl);...
                sum(vl2),sum(l2),sum(vl),sum(l);...
                sum(v2l),sum(vl),sum(v2),sum(v);...

sum(vl),sum(l),sum(v),length])*[sum(vltsi);sum(ltsi);sum(vtsi);sum(tsi)];
        a1=a(1);
        a2=a(2);
        a3=a(3);
        a4=a(4);

        b=inv([sum(v2r2),sum(v2r),sum(vr2),sum(vr);...
                sum(v2r),sum(v2),sum(vr),sum(v);...
                sum(vr2),sum(vr),sum(r2),sum(r);...

sum(vr),sum(v),sum(r),length])*[sum(vrtso);sum(vtso);sum(rtso);sum(tso)];
        b1=b(1);
        b2=b(2);
        b3=b(3);
        b4=b(4);


        rvar=(m+.01:200);
        vvar=(2:.1:20);

        for n1=1:max(size(rvar));
            for n2=1:max(size(vvar));
```

```
169                 tsisurf(n2,n1)=a1*vvar(n2)*log(rvar(n1)-
    m)+a2*log(rvar(n1))+a3*vvar(n2)+a4;
170                 tsosurf(n2,n1)=b1*vvar(n2)*log(rvar(n1)-
    m)+b2*log(rvar(n1))+b3*vvar(n2)+b4;
171             end;
172         end;
173 
174         for n=1:length;
175             ei2(n)=abs(tsi(n)-a1*v(n)*log(r(n)-m)-a2*log(r(n)-m)-a3*v(n)-a4);
176             eo2(n)=abs(tso(n)-b1*v(n)*log(r(n)-m)-b2*log(r(n)-m)-b3*v(n)-b4);
177         end;
178 
179         if size(ei2,1) == 1;
180             ei2=ei2';
181         end;
182         if size(eo2,1) == 1;
183             eo2=eo2';
184         end;
185 
186         figure('Color','w','Position',[0 0 1024 768]);
187         movegui('center');
188         subplot(2,2,2);
189         points=plot3(r,v,tsi,'b*');
190         hold on;
191         crv=surf(rvar,vvar,tsisurf,'EdgeColor','none');
192         hold off;
193         xlabel('Radius of Curvature (ft.)');
194         ylabel('Velocity (ft/s)');
195         zlabel('Trackslip');
196         title(['Track Slip 3D Plots']);
197         subplot(2,2,1);
198         set(gca,'Visible','off');
199         text(.5,.8,...
200             ['\bfInner (' intrkside ') Track Slip Data for ' turndir],...
201             'HorizontalAlignment','center');
202         for n=2:4;
203             if a(n)>=0
204                 snl(n-1)='+';
205             else
206                 snl(n-1)='-';
207             end;
208         end;
209         text(.5,.6,...
210             ['TS = ' num2str(abs(a1)) '\nuln(\it(r)-' num2str(m) ')'...
211                 snl(1) num2str(abs(a2)) 'ln(\it(r)-' num2str(m) ')' snl(2)...
212                 num2str(abs(a3)) '\nu' snl(3)
    num2str(abs(a4))],'HorizontalAlignment','center');
213 
214 
215 
216 
217     %     legend(['TS=' num2str(abs(a1)) '\nuln((\itr)-' num2str(m) ')' snl(1)
    num2str(abs(a2))...
218     %             'ln((\itr)-' num2str(m) ')' snl(2) num2str(abs(a3)) '\nu' snl(3)
    num2str(abs(a4))],0);
219 
220     %     figure('Color','w');
221         subplot(2,2,3);
222         stem(r,ei2,'-bs');
223         % eps2=mean((tsl-a1.*v.*log(r)+a2.*log(r)+a3.*v+a4).^2);
224         xlabel('Radius of Curvature (ft.)');
225         ylabel('(\epsilon)');
226         title(['Error Plot Compared to Radius of Curvature']);
227         subplot(2,2,4);
228         stem(v,ei2,'-bs');
229         % eps2=mean((tsl-a1.*v.*log(r)+a2.*log(r)+a3.*v+a4).^2);
230         xlabel('Velocity (ft/s)');
231         ylabel('(\epsilon)');
232         title(['Error Plot Compared to Velocity']);
233 
234         figure('Color','w','Position',[0 0 1024 768]);
```

```
235         movegui('center');
236         subplot(2,2,2);
237         points=plot3(r,v,tso,'b*');
238         hold on;
239         crv=surf(rvar,vvar,tsosurf,'EdgeColor','none');
240         hold off;
241         xlabel('Radius of Curvature (ft.)');
242         ylabel('Velocity (ft/s)');
243         zlabel('Trackslip');
244         title(['Track Slip 3D Plots']);
245         subplot(2,2,1);
246         set(gca,'Visible','off');
247         text(.5,.8,...
248             ['\bfOuter (' outtrkside ') Track Slip Data for ' turndir],...
249             'HorizontalAlignment','center');
250 
251         for n=2:4;
252             if b(n)>=0
253                 snr(n-1)='+';
254             else
255                 snr(n-1)='-';
256             end;
257         end;
258         text(.5,.6,...
259             ['TS = ' num2str(abs(b1)) '\nu\it(r)'...
260                 snr(1) num2str(abs(b2)) '\nu' snl(2)...
261                 num2str(abs(b3)) '\it(r)' snr(3)
    num2str(abs(b4))],'HorizontalAlignment','center');
262 
263     %     legend(['TS=' num2str(b1) '\nuln[\itr]' snr(1) num2str(abs(b2))...
264     %             'ln[\itr]' snr(2) num2str(abs(b3)) '\nu' snr(3)
    num2str(abs(b4))],0);
265 
266     %     figure('Color','w');
267         subplot(2,2,3);
268         stem(r,eo2,'-bs');
269         % eps2=mean((tsl-a1.*v.*log(r)+a2.*log(r)+a3.*v+a4).^2);
270         xlabel('Radius of Curvature (ft.)');
271         ylabel('(\epsilon)');
272         title(['Error Plot Compared to Radius of Curvature']);
273         subplot(2,2,4);
274         stem(v,eo2,'-bs');
275         % eps2=mean((tsl-a1.*v.*log(r)+a2.*log(r)+a3.*v+a4).^2);
276         xlabel('Velocity (ft/s)');
277         ylabel('(\epsilon)');
278         title(['Error Plot Compared to Velocity']);
279     end;
```